

FreeDOS

by Jim Hall and Kevin O'Brien

We are Opensource.com

Opensource.com is a community website publishing stories about creating, adopting, and sharing open source solutions. Visit Opensource.com to learn more about how the open source way is improving technologies, education, business, government, health, law, entertainment, humanitarian efforts, and more.

Do you have an open source story to tell? Submit a story idea at opensource.com/story

Email us at open@opensource.com



Supported by
Red Hat

Table of Contents

Get started with FreeDOS.....	5
A brief history of FreeDOS.....	11
How a college student founded a free and open source operating system.....	16
Install and remove software packages on FreeDOS.....	23
Install FreeDOS without the installer.....	33
How to use FreeDOS as an embedded system.....	44
FreeDOS commands you need to know.....	53
Understanding file names and directories in FreeDOS.....	57
How to navigate FreeDOS with CD and DIR.....	61
FreeDOS commands for Linux fans.....	66
Set your path in FreeDOS.....	72
Appendix: Navigate your FreeDOS system.....	76

Jim Hall



At

Jim Hall is an open source software advocate and developer, best known for usability testing in GNOME and as the founder + project coordinator of FreeDOS. In his current work, Jim is CEO of Hallmentum, an IT executive consulting company that provides hands-on IT Leadership training, workshops, and coaching.

Kevin O'Brien



Kevin is a former Project Manager at Ford Motor Credit Company, a contributor to Hacker Public Radio and Full Circle Magazine, the former Publicity Director for Ohio Linux Fest, former Tech Track manager for Penguicon, and now very happily retired.

Get started with FreeDOS

By Jim Hall

Throughout the 1980s and into the 1990s, I was primarily a DOS user. I loved the command line environment offered in DOS, which became more powerful with each successive release. I even learned how to write my own DOS programs in the C programming language so I could extend the DOS command line, and write more powerful replacements for the standard DOS commands. I'd experimented with Microsoft's Windows—but if you remember Windows 3 from that time, you know it was slow and tended to crash. But I preferred the command line anyway, so I stuck to DOS.

That all changed in 1994. Popular tech magazines talked about an upcoming version of Windows that would completely do away with DOS. I didn't want to be forced to Windows. On the discussion boards I visited on Usenet, others felt the same. So [on 29 June 1994](#), I decided that if we wanted to keep DOS, we needed to write our own. So on June 29, I announced a small project that would become [The FreeDOS Project](#).

Since then, we've released several full distributions of FreeDOS. We started with the alpha series from 1994 to 1997, the beta series from 1998 to 2005, before finally releasing the FreeDOS 1.0 distribution in 2006. Progress has been slow but steady since then. We haven't really been rushed to release each new version after 1.0, because DOS stopped being a moving target in 1995.

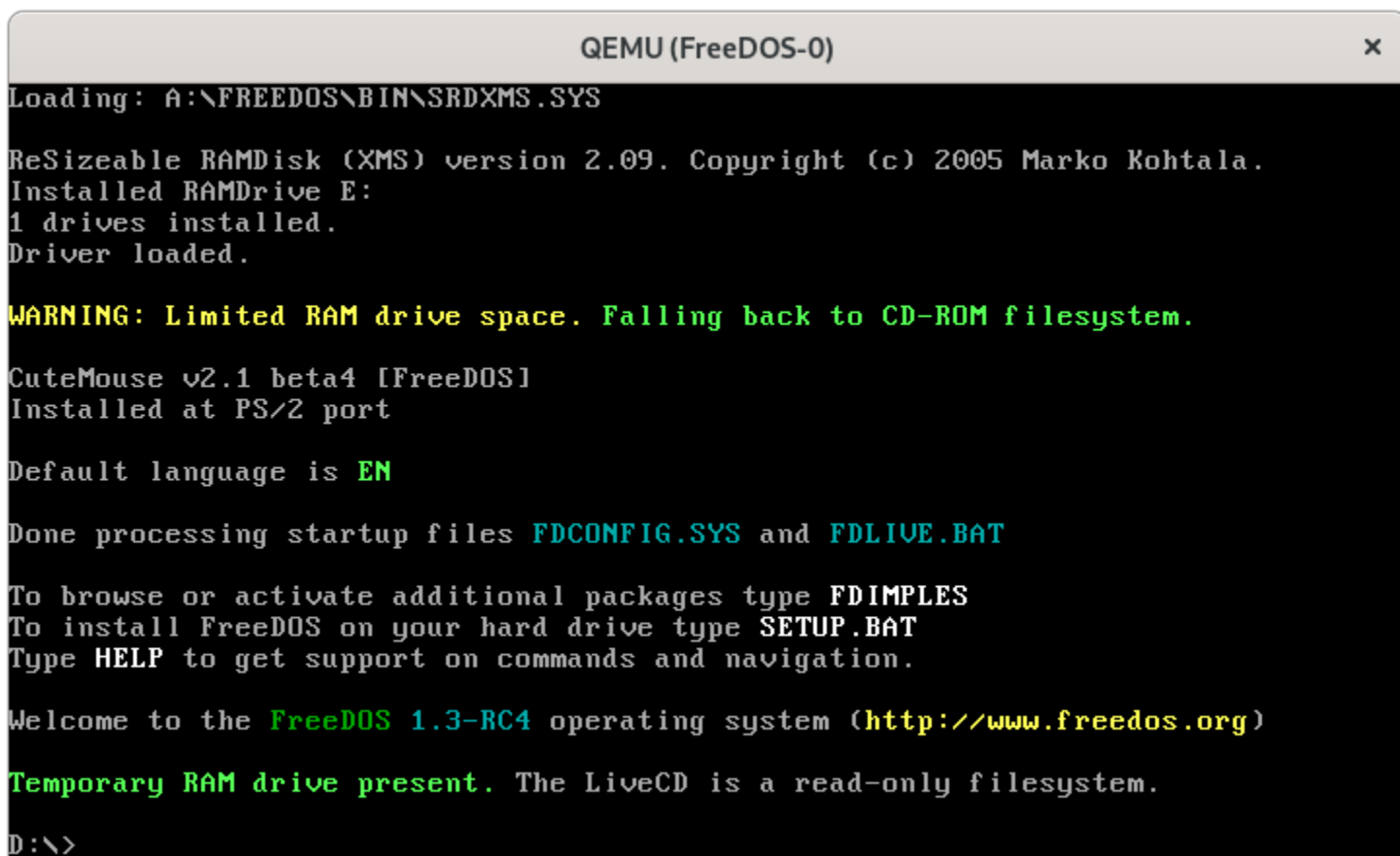
Each FreeDOS distribution since 1.0 has been a continual re-imagining of what a modern DOS might look like. We've included lots of compilers, assemblers for developers to write software. We also provide lots of "power tools" so you can do real work. And we offer a variety of editors because everyone has their favorite.

We recently released the FreeDOS distribution. This is technically a release candidate towards our upcoming FreeDOS distribution, but it's a full-featured distribution. I'm very excited about all the great features in FreeDOS.

Run FreeDOS without installing FreeDOS

In all our previous FreeDOS distributions, we focused on *installing* FreeDOS to a computer. But we recognize that most users don't actually run FreeDOS on actual hardware anymore—they run FreeDOS in [a virtual machine like QEMU or VirtualBox](#). So in FreeDOS, we improved the "LiveCD" environment.

With FreeDOS, you can just boot the LiveCD image in your favorite virtual machine, and start using FreeDOS right away. That's how I run FreeDOS now; I have a small virtual hard drive image where I store all my files, but I boot and run FreeDOS from the LiveCD.



```
QEMU (FreeDOS-0)
Loading: A:\FREEDOS\BIN\SRDXMS.SYS
ReSizeable RAMDisk (XMS) version 2.09. Copyright (c) 2005 Marko Kohtala.
Installed RAMDrive E:
1 drives installed.
Driver loaded.
WARNING: Limited RAM drive space. Falling back to CD-ROM filesystem.
CuteMouse v2.1 beta4 [FreeDOS]
Installed at PS/2 port
Default language is EN
Done processing startup files FDCONFIG.SYS and FDLIVE.BAT
To browse or activate additional packages type FDIMPLES
To install FreeDOS on your hard drive type SETUP.BAT
Type HELP to get support on commands and navigation.
Welcome to the FreeDOS 1.3-RC4 operating system (http://www.freedos.org)
Temporary RAM drive present. The LiveCD is a read-only filesystem.
D:\>
```

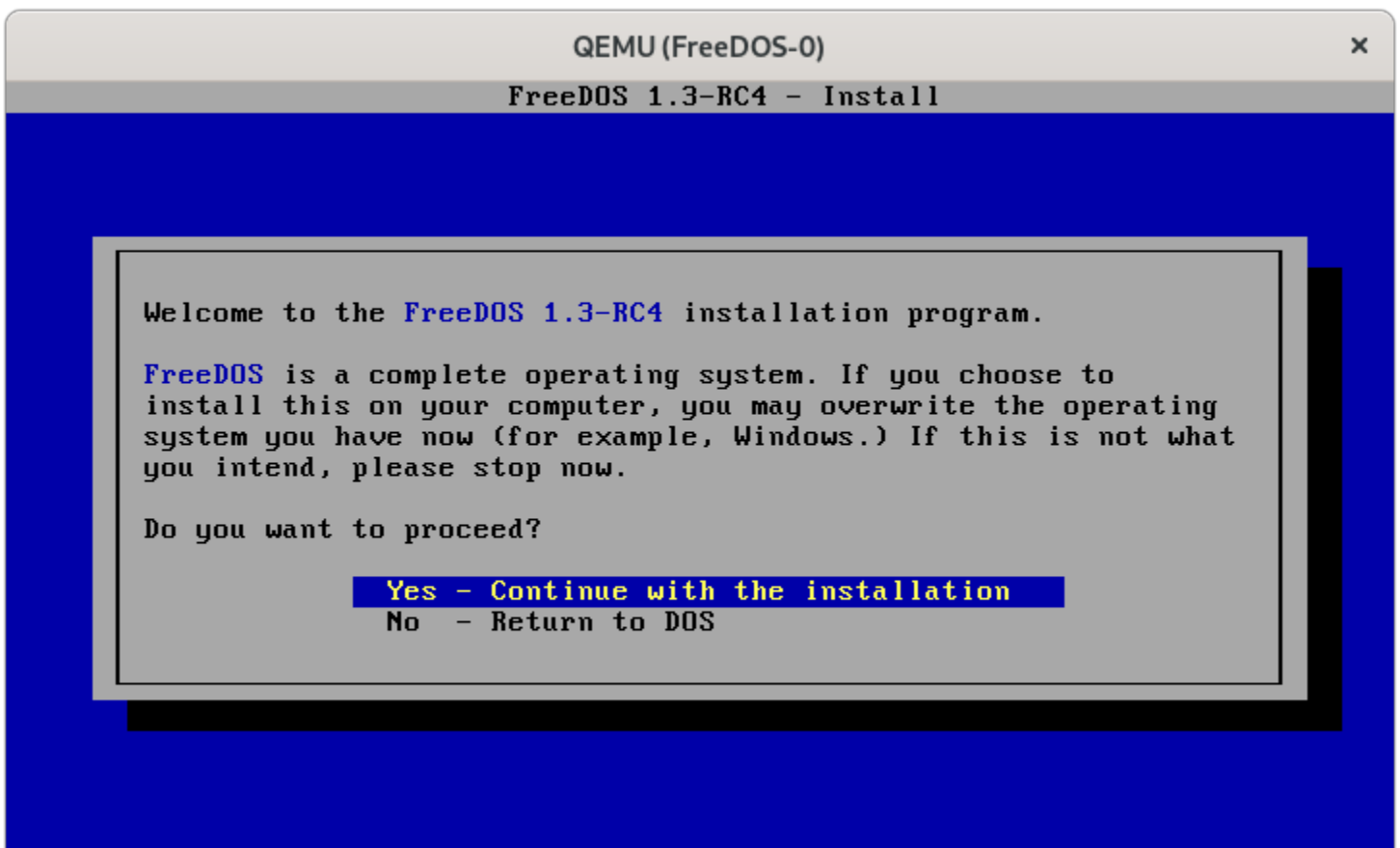
Booting the FreeDOS LiveCD

Installing is really easy

If you don't want to run FreeDOS from the LiveCD, you can also install it on your hard drive. We updated the installer in FreeDOS so it's not really a "program" per se, but instead is a very

smart DOS "batch" file that detects all sorts of things and takes the appropriate action, like creating a new disk partition for FreeDOS if none exist already.

Older FreeDOS distributions used to prompt you for everything, even selecting individual programs to install. The new installer is very streamlined. It asks you a few questions to get started, then does everything else on its own. Installing FreeDOS on an empty virtual machine takes only a few minutes.



Installing FreeDOS

You can install it from floppy

Not everyone prefers to run FreeDOS in a virtual machine. There's a retrocomputing community out there that collects and lovingly restores classic PC hardware like Pentium or '486 systems. You can even find some XT (8088) or AT (80286) systems out there, kept running by a dedicated user community.

And while we consider FreeDOS a *modern* DOS, we wouldn't be "DOS" if we didn't also run on the older PC hardware too. So with FreeDOS 1.3, we include a Floppy-Only Edition! This edition should run on any hardware that can run FreeDOS and has EGA or better graphics.

Are you running a '286 or another classic system without a CD-ROM drive? Install from these floppies to install FreeDOS. Do you have just one hard drive and no CD or floppy drive? Just copy the contents of the floppies to a temporary directory and run the installer from there. Want to perform a "headless" install to a different DOS directory? It's easy with the command-line options.

The Floppy-Only Edition uses a completely different installer and contains a limited FreeDOS set of programs that are more useful on classic PC hardware.



Installing the FreeDOS Floppy-Only Edition

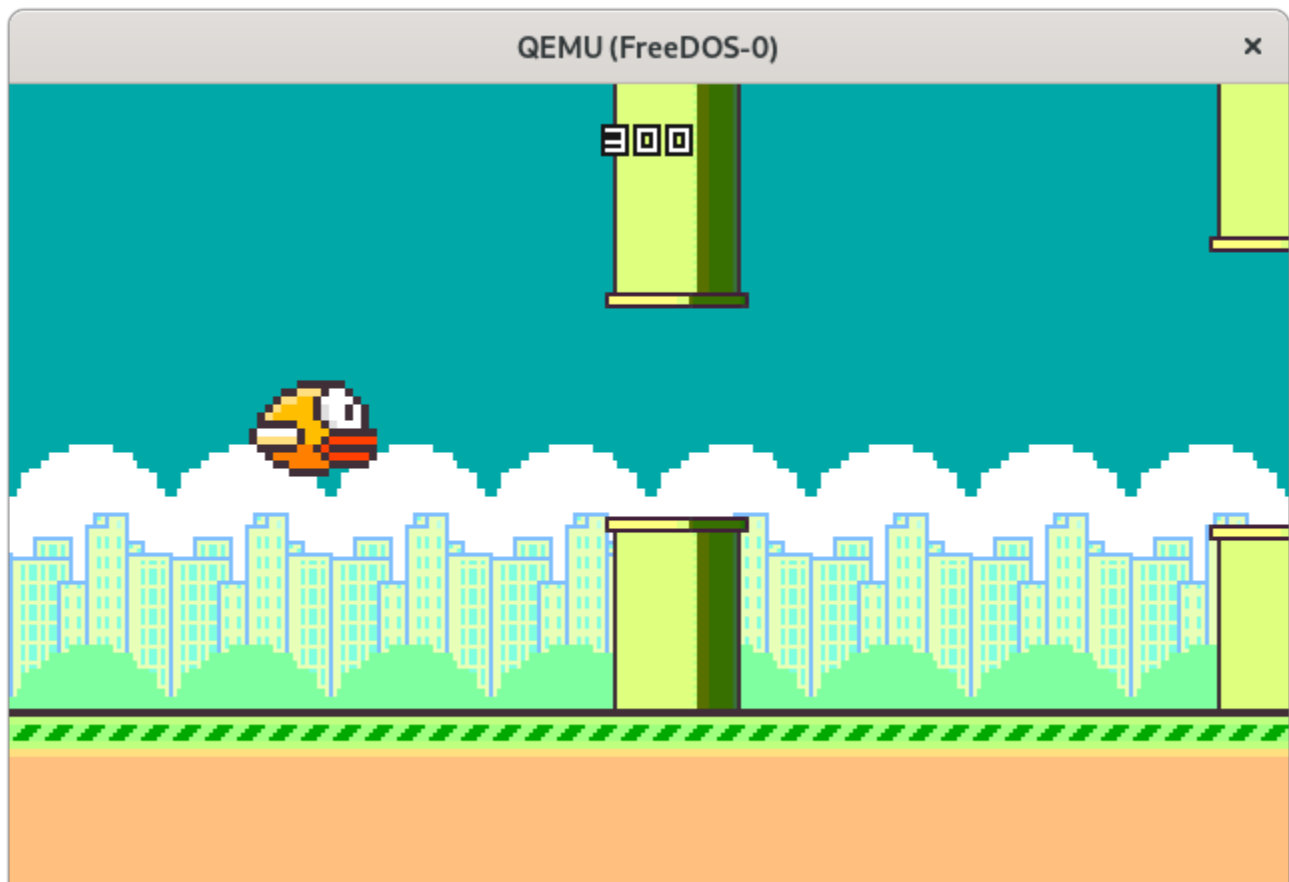
Filled with open source apps and games

FreeDOS isn't a *free* DOS if it's a closed source DOS. We want everyone to be able to use and study FreeDOS, including its source code. As we planned the FreeDOS distribution, we took a

close look at every license in every package and focused on including only *open source* programs. (A few programs in previous FreeDOS distributions were not quite "open source," and one or two programs didn't include source code but were otherwise "free to use and distribute." In this release, everything is open source, using the Open Source Definition as our model.)

And what a great collection of open source apps and games. The games are my favorite addition to FreeDOS. Many people use FreeDOS to play classic DOS games, but we wanted to provide our own open source games for people to play.

You can find two games already installed in the LiveCD: Simple Senet (a board game dating to ancient Egypt) and Floppy Bird (a version of the Flappy Bird game). If you install FreeDOS, you'll also find lots of other games to try, including Sudoku86 (a sudoku game), Wing (a space shooter), and Bolitaire (solitaire card game).



Playing the Floppy Bird game



The ancient game of Senet

Try FreeDOS now

You can find the new FreeDOS from the FreeDOS website, on our [Downloads](#) page. To install FreeDOS, you'll need at least 20MB of free disk space: 20MB to install a plain FreeDOS system, or 250MB to install everything, including applications and games. To install the source code too, you'll need up to 450MB of free space.

A brief history of FreeDOS

By Jim Hall

A master was explaining the nature of [The Tao of Programming](#) to one of his novices. "The Tao is embodied in all software—regardless of how insignificant," said the master.

"Is Tao in a hand-held calculator?" asked the novice.

"It is," came the reply.

"Is the Tao in a video game?" continued the novice.

"It is even in a video game," said the master.

"And is the Tao in the DOS for a personal computer?"

The master coughed and shifted his position slightly.

"The lesson is over for today," he said.

The Tao of Programming, Geoffrey James, InfoBooks, 1987

Computing used to be limited only to expensive mainframes and "Big Iron" computer systems like the PDP11. But the advent of the microprocessor brought about a computing revolution in the 1970s. You could finally have a computer in your home—the "personal computer" had arrived!

The earliest personal computers I remember seeing included the Commodore, TRS-80, and Apple. The personal computer became such a hot topic that IBM decided to enter the market. After a rapid development cycle, IBM released the IBM 5150 Personal Computer (the original "IBM PC") in August 1981.

Creating a computer from scratch is no easy task, so IBM famously used "off-the-shelf" hardware to build the PC, and licensed other components from outside developers. One of those was the operating system, licensed from Microsoft. In turn, Microsoft acquired 86-DOS

from Seattle Computer Products, applied various updates, and debuted the new version with the IBM PC as IBM PC-DOS.

Early DOS

Running in memory *up to* 640 kilobytes, DOS really couldn't do much more than manage the hardware and allow the user to launch applications. As a result, the PC-DOS 1.0 command line was pretty anæmic, only including a few commands to set the date and time, manage files, control the terminal, and format floppy disks. DOS also included a BASIC language interpreter, which was a standard feature in all personal computers of the era.

It wasn't until PC-DOS 2.0 that DOS became more interesting, adding new commands to the command line, and including other useful tools. But for me, it wasn't until MS-DOS 5.0 in 1991 that DOS began to feel "modern." Microsoft overhauled DOS in this release, updating many of the commands and replacing the venerable Edlin editor with a new full-screen editor that was more user-friendly. DOS 5 included other features that I liked, as well, such as a new BASIC interpreter based on Microsoft QuickBASIC Compiler, simply called QBASIC. If you've ever played the Gorillas game on DOS, it was probably in MS-DOS 5.0.

Despite these upgrades, I wasn't entirely satisfied with the DOS command line. DOS never strayed far from the original design, which proved limiting. DOS gave the user a few tools to do some things from the command line—otherwise, you were meant to use the DOS command line to launch applications. Microsoft assumed the user would spend most of their time in a few key applications, such as a word processor or spreadsheet.

But developers wanted a more functional DOS, and a sub-industry sprouted to offer neat tools and programs. Some were full-screen applications, but many were command-line utilities that enhanced the DOS command environment. When I learned a bit of C programming, I started writing my own utilities that extended or replaced the DOS command line. And despite the rather limited underpinnings of MS-DOS, I found that the third-party utilities, plus my own, created a powerful DOS command line.

FreeDOS

In early 1994, I started seeing a lot of interviews with Microsoft executives in tech magazines saying the next version of Windows would totally do away with DOS. I'd used Windows before—but if you remember the era, you know Windows 3.1 wasn't a great platform. Windows 3.1 was clunky and buggy—if an application crashed, it might take down the entire Windows system.

And I didn't like the Windows graphical user interface, either. I preferred doing my work at the command line, not with a mouse.

I considered Windows and decided, If Windows 3.2 or Windows 4.0 will be anything like Windows 3.1, I want nothing to do with it. But what were my options? I'd already experimented with Linux at this point, and thought [Linux was great](#)—but Linux didn't have any applications. My word processor, spreadsheet, and other programs were on DOS. I needed DOS.

Then I had an idea! I thought, If developers can come together over the internet to write a complete Unix operating system, surely we can do the same thing with DOS. After all, DOS was a fairly straightforward operating system compared to Unix. DOS ran one task at a time (single-tasking) and had a simpler memory model. It shouldn't be *that* hard to write our own DOS.

So on June 29, 1994, I [posted an announcement](#) to `comp.os.msdos.apps`, on a message board network called Usenet:

ANNOUNCEMENT OF PD-DOS PROJECT:

A few months ago, I posted articles relating to starting a public domain version of DOS. The general support for this at the time was strong, and many people agreed with the statement, "start writing!" So, I have...

Announcing the first effort to produce a PD-DOS. I have written up a "manifest" describing the goals of such a project and an outline of the work, as well as a "task list" that shows exactly what needs to be written. I'll post those here, and let discussion follow.

A note about the name—I wanted this new DOS to be something that everyone could use, and I naively assumed that when everyone could use it, it was "public domain." I quickly realized the difference, and we renamed "PD-DOS" to "Free-DOS"—and later dropped the hyphen to become "FreeDOS."

A few developers reached out to me, to offer utilities they had created to replace or enhance the DOS command line, similar to my own efforts. We pooled our utilities and created a useful system that we released as "Alpha 1" in September 1994, just a few months after announcing the project. Development was pretty swift in those days, and we followed up with "Alpha 2" in December 1994, "Alpha 3" in January 1995, and "Alpha 4" in June 1995.

A modern DOS

Since then, we've always focused on making FreeDOS a "modern" DOS. And much of that modernization is centered on creating a rich command-line environment. Yes, DOS still needs to support applications, but we believe FreeDOS needs a strong command-line environment, as well. That's why FreeDOS includes dozens of useful tools, including commands to navigate directories, manage files, play music, connect to networks, and a collection of Unix-like utilities such as `less`, `du`, `head`, `tail`, `sed`, and `tr`.

While FreeDOS development has slowed, it has not stopped. Developers continue to write new programs for FreeDOS, and add new features to FreeDOS. I'm particularly excited about several great additions to FreeDOS, the latest release candidate for the forthcoming FreeDOS 1.3. A few recent updates:

- Mateusz Viste created a new ebook reader called Ancient Machine Book (AMB) that we've leveraged as the new help system in FreeDOS
- Rask Ingemann Lambertsen, Andrew Jenner, TK Chia, and others are updating the IA-16 version of GCC, including a new *libi86* library that provides some degree of compatibility with the Borland Turbo C++ compiler's C library
- Jason Hood has updated an unloadable CD-ROM redirector substitute for Microsoft's MSCDEX, supporting up to 10 drives
- Superllu has created DOjS, a Javascript development canvas with an integrated editor, graphics and sound output, and mouse, keyboard, and joystick input
- Japheth has created a DOS32PAE extender that is able to use huge amounts of memory through PAE paging

Despite all of the new development on FreeDOS, we remain true to our DOS roots. As we continue working toward FreeDOS"final," we carry several core assumptions, including:

- **Compatibility is key**—FreeDOS isn't really "DOS" if it can't run classic DOS applications. While we provide many great open source tools, applications, and games, you can run your legacy DOS applications, too.
- **Continue to run on old PCs (XT, '286, '386, etc)**—FreeDOS will remain 16-bit Intel but will support new hardware with expanded driver support, where possible. For this reason, we continue to focus on a single-user command-line environment.
- **FreeDOS is open source software**—I've always said that FreeDOS isn't a "free DOS" if people can't access, study, and modify the source code. FreeDOS will include software that uses recognized open source licenses as much as possible. But DOS

actually pre-dates the GNU General Public License (1989) and the Open Source Definition (1998) so some DOS software might use its own "free with source code" license that isn't a standard "open source" license. As we consider packages to include in FreeDOS, we continue to evaluate any licenses to ensure they are suitably "open source," even if they are not officially recognized.

We welcome your help in making FreeDOS great! Please join us on our email list—we welcome all newcomers and contributors. We communicate over an email list, but the list is fairly low volume so is unlikely to fill up your Inbox.

Visit the FreeDOS website at www.freedos.org.

How a college student founded a free and open source operating system

By Opensource.com

[Jim Hall](#) is best known as the computer programmer who founded the FreeDOS project. Jim began the project in 1994 as a replacement for MS-DOS while he was still a student at the University of Wisconsin–River Falls. Jim created FreeDOS in response to Microsoft ending support for MS-DOS in 1994. Recently Jim agreed to an email interview. Correspondent Joshua Allen Holm joined me in posing the following questions to Jim.

Don Watkins: What kind of skill set invites you to write your own operating system?

I think even a beginner can get started writing an operating system like FreeDOS, although it would take a more advanced programmer to write the kernel.

[I am a self-taught programmer.](#) I learned about programming from an early age by tinkering on our Apple II computer at home. Much later, I learned C programming—my brother was a computer science student when I was a physics student, and he introduced me to C. I picked up the rest by reading books and writing my own programs.

I wrote a lot of small utilities that enhanced my command line on MS-DOS or even replaced certain DOS commands. And you can write a lot of those programs even with a basic level of programming experience. You can write file utilities like FIND, FC, CHOICE, TYPE, MORE, or COPY—or user commands like ECHO or CLS—with only an introduction to C programming. With a bit of practice, you can write system-level programs like ATTRIB, or the COMMAND shell.

Don Watkins: Were you inspired by Linus Torvalds when you decided to write your own version of DOS and how did that contribute to your licensing decision?

In a way, yes. I really liked DOS and had been using it since the early 1980s. I ran MS-DOS on my personal computer at university. But in 1993, I discovered Linux.

I really liked the Unix systems in our campus computer lab, where I spent much of my time as an undergraduate university student. When I heard about Linux, a free version of Unix that I could run on my '386 computer at home, I immediately wanted to try it out. [My first Linux distribution](#) was Softlanding Linux System (SLS) 1.03, with Linux kernel 0.99 alpha patch level 11. That required a whopping 2MB of RAM, or 4MB if you wanted to compile programs, and 8MB to run X windows.

I dual-booted Linux with MS-DOS. I booted into Linux most of the time, but I still booted back into MS-DOS to write papers in a word processor, to use a spreadsheet program to analyze data for my lab classes or to play my favorite DOS games.

In 1994, I heard that Microsoft planned to “do away” with MS-DOS. The next version of Windows would eliminate DOS. I didn’t like that, and I still wanted to run DOS. I decided that if folks could come together over the Internet to write something like Linux, surely, we could do the same with DOS. After all, DOS was fairly simple compared to Linux.

On June 29, 1994, [I announced the “PD-DOS” project](#) to write our own DOS. I called it “PD” because I thought it would be in the public domain. But I quickly learned about the GNU General Public License that the Linux kernel used, and decided that was a much better license. No one could take our source code and create a proprietary version of DOS. We changed the name to “Free-DOS” after another week or so. We later dropped the hyphen to become “FreeDOS.”

Joshua Allen Holm: What are the advantages of using FreeDOS over alternative ways of running DOS applications (for example, DOSBox)?

Using DOSBox to run DOS applications in Linux is a great way to run certain DOS applications. But DOSBox is really intended to launch a single DOS program, like a game. The DOS command line is pretty limited in DOSBox.

In contrast, FreeDOS provides a full DOS command line. We include all of the commands you remember from classic DOS, and added other commands and utilities to do new things. FreeDOS also includes compilers and assemblers so developers can write new programs,

utilities, and tools to make your DOS experience more useful, Internet programs to help you get on a network, and even open source games.

Joshua Allen Holm: Looking back over the years you have worked on FreeDOS, is there anything you would have done differently?

There's only one event that I wish I could take back. I occasionally reach out to companies that sold DOS applications in the 1980s and 1990s and ask them if they will release the source code to their old DOS programs under an open source software license. That's a great way to contribute to the open source community.

One popular DOS program was a replacement for the MS-DOS COMMAND shell. 4DOS, by JP Software, was an extremely powerful DOS shell and included many modern features. For example, 4DOS supported built-in aliases, color-coded directory listings, and a "swapping" mechanism that freed up more conventional memory to run programs.

I contacted JP Software to ask if they would release the source code to 4DOS under an open source software license. JP Software had stopped supporting DOS, and instead focused on a similar replacement for the CMD shell in Windows NT, called 4NT. They were interested in releasing the source code to 4DOS but were concerned that someone might take the 4DOS source code and release a version for Windows. In effect, that would put JP Software in competition with their older product.

I still didn't understand the fine points of open source software licenses, and I gave them bad advice. I suggested they might start with an existing open source license and add a term that said you could only run it on DOS. They then released the 4DOS source code under a modified version of the MIT license.

Unfortunately, limiting where you can run the software violates one of the tenets of open source software and free software. Users should be able to run open source software anywhere, and for any use. An open source license isn't "open source" if you are limited to running it only on one operating system.

So despite best intentions, I gave JP Software really bad advice there, and 4DOS isn't actually open source software. We used to include 4DOS in FreeDOS—but as we are preparing the FreeDOS release, we want to be careful to only include open source software. So FreeDOS ("release candidate 4") does not include 4DOS.

Joshua Allen Holm: What are some interesting ways people are using FreeDOS?

Over the years, I've seen people use FreeDOS to do a lot of really interesting things!

One of the earliest cool examples was someone who built pinball machines like you used to see in arcades. He embedded a version of FreeDOS to track the score and update the video screen on the back of the machine. I don't know exactly how he did this, but my guess is every target or bumper on the pinball board probably generated a keyboard event. You can write a DOS application to read the "keyboard" and update the score based on that.

A few years ago, a user found a video of a train control system in Russia [that ran on a FreeDOS PC](#). They rebooted the computer, and if you freeze the video at the right point, you can briefly see the FreeDOS kernel starting up. It disappears quickly, but you can see it at 0:07 in the video.

More recently, I saw someone had managed to boot an original IBM PC 5150 with FreeDOS [from a vinyl record](#), using the 5150's rarely used cassette tape storage port. It's really cool to see FreeDOS being used this way. It's a method that I would never have thought to try, but sometimes you have to do something just for the fun of it.

Joshua Allen Holm: Why work on DOS in 2021?

We still work on DOS in 2021 for a few reasons. I guess the first reason is that DOS is still interesting. We've added a lot to FreeDOS over the years. Where the original MS-DOS had a limited set of commands, FreeDOS includes dozens of useful utilities and tools, including editors, compilers, assemblers, games, and other neat programs.

But it has to be more than just a cool hobby. I find that working on FreeDOS makes for a very interesting programming challenge. In modern systems like Linux, you can take advantage of a lot of memory at once, and you can address it all in one big block. As a result, many programmers will load a lot of libraries and other code to create their projects. This is a very easy way to build a complicated project. You can build a very complex system in a very short time this way. And for many systems, time to market is the most important factor.

Loading a bunch of libraries and other code blocks is very inefficient, however. You may have the same basic functionality implemented half a dozen ways across the different libraries because each library implements something their own way. So your code grows and requires more memory.

Maybe that's not a problem on a desktop PC. I run Linux, and my modern desktop PC has 32GB of memory. Loading a bunch of stuff into memory isn't a big deal. But on a shared server, where you might have multiple instances of that project running, you'll quickly run into

memory limitations. How many instances can you run at the same time on a server? That 32GB of memory starts to look pretty slim.

You can't load all of that into memory on a DOS machine. To remain compatible with the original DOS, FreeDOS has all the limitations of DOS. When MS-DOS was popular, a powerful PC might have had 4MB, 8MB, or even 16MB of extended memory. But the computer only had 640kb of "main" memory, due to how DOS addressed memory. And that's megabytes and kilobytes, not gigabytes. A kilobyte is a thousand bytes (the basic unit of memory). A megabyte is a thousand kilobytes. And a gigabyte is a thousand megabytes. So today's computers have memory that is about 1,000,000 more than a DOS computer.

By programming on a limited system like FreeDOS, you constantly have to think about the tradeoffs. How much memory does my program really need to do its job? Is it faster to read a file into memory to work on it, or process the file one bit at a time? And you're always keeping in mind what libraries and other code you use in your program. A DOS program can only be so big, so you need to be careful about how you write a DOS program.

When you write DOS programs all the time, you get really good at optimizing a program. You think about programming in a different way, because you're always considering how to do something more efficiently. That's a challenge, but an interesting one.

Don Watkins: How big is the FreeDOS community?

FreeDOS was a very popular project throughout the 1990s and into the early 2000s, but the community isn't as big these days. But it's great that we are still an engaged and active group. If you look at the news items on our website, you'll see we post updates on a fairly regular basis.

It's hard to estimate the size of the community. I'd say we have a few dozen members who are very active. And we have a few dozen others who reappear occasionally to post new versions of their programs. I think to maintain an active community that's still working on an open source DOS from 1994 is a great sign.

Some members have been with us from the very beginning, and I'm really thankful to count them as friends. We do [video hangouts on a semi-regular basis](#). It's great to finally "meet" the folks I've only exchanged emails with over the years.

It's meetings like this when I remember open source is more than just writing code; it's about a community. And while I've always done well with our virtual community that communicates via

email, I really appreciated getting to talk to people without the asynchronous delay or artificial filter of email—making that real-time connection means a lot to me.

Don Watkins: How does someone get involved in the community?

I think our community is very welcoming, so anyone is free to join. We communicate via an email list, which you can find on the [FreeDOS website](#). Join the freedos-user email list if you want to talk about FreeDOS or ask for help. Developers should join the freedos-devel email list; that's where most of the FreeDOS developers hang out. Our email list volume is pretty low, so you aren't likely to fill up your inbox by subscribing to either email list.

A great way to get started is by writing or updating documentation, or by fixing bugs. I think that's true of pretty much every open source project out there. We always need folks to work on the documentation and fix bugs. But for a project like FreeDOS, I think reading through the documentation is important if you're new to DOS. A common mistake for newcomers is thinking of FreeDOS as a stripped-down version of Linux, when in fact DOS uses a different memory and execution model. You can learn about that by reading the documentation, which is why I recommend new contributors start there.

For the more adventurous, we maintain a list of priority projects on our website. If you'd like to contribute to FreeDOS, but aren't sure what needs work, you might consider tackling one or more of these projects:

If you've got some programming experience:

- Port FreeDOS utilities to OpenWatcom C and NASM—our preferred C compiler and Assembler for FreeDOS. (Some older FreeDOS programs were probably written for Borland C or Turbo C or Microsoft C, or Microsoft ASM or Turbo ASM)
- Port GNU utilities to FreeDOS, such as using IA-16 GCC (while IA-16 GCC requires a '386 or better to compile, programs compiled with IA-16 GCC run on all CPUs)
- Create a new alternative shell, similar to COMMAND.COM but with expanded BAT programming
- Add international language support to a FreeDOS program that currently only supports one language.

If you're a highly-skilled DOS developer:

- Write a guest tool like VMSMOUNT for VirtualBox

- Write a driver for modern sound cards
- Add some kind of UEFI bootstrap BIOS emulator, perhaps implemented as a CSM

We like to make it easy for new contributors to get started in FreeDOS, and we welcome everyone who wants to work on FreeDOS. If you still don't know how to contribute, feel free to ask on the email list.

Install and remove software packages on FreeDOS

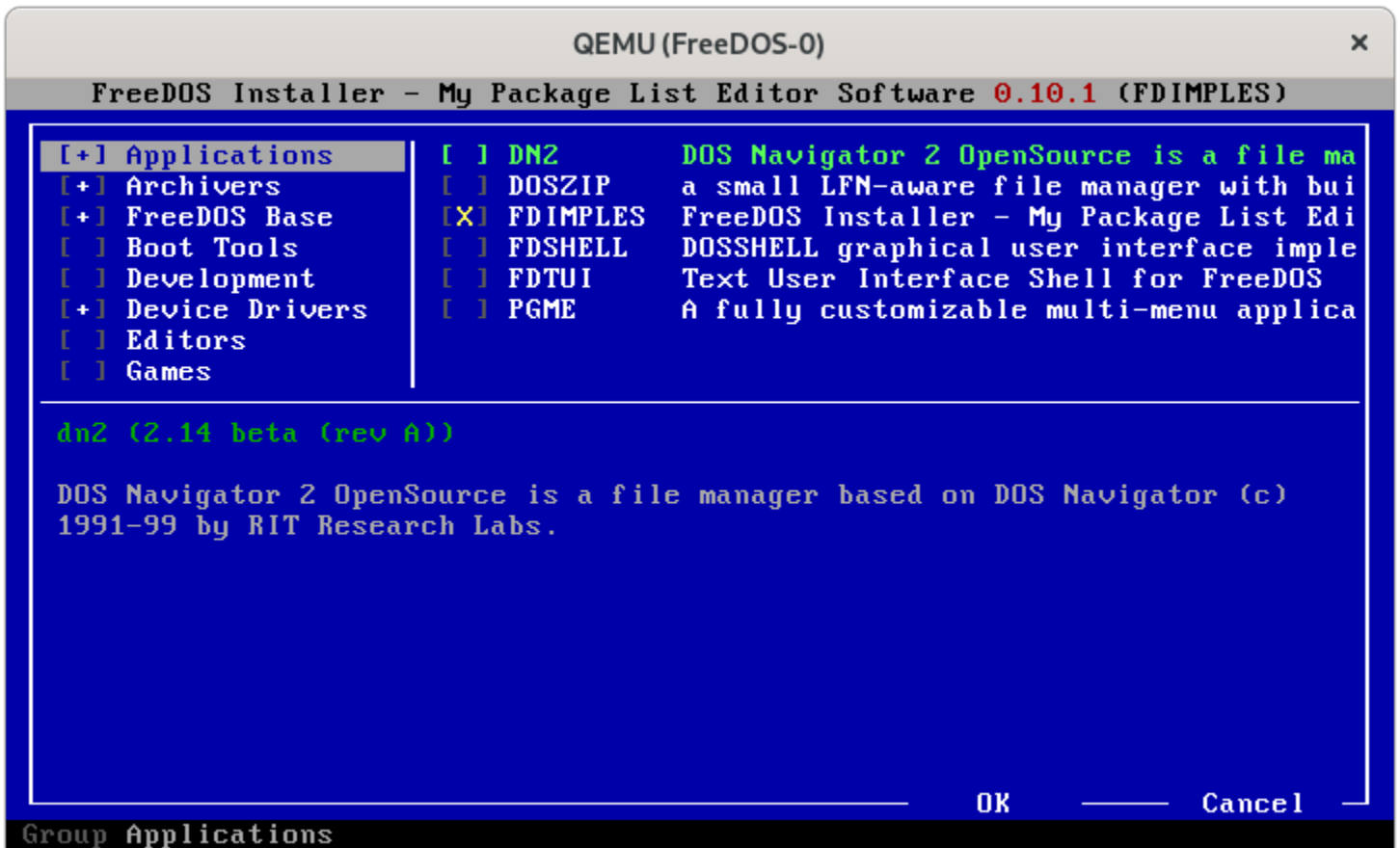
By Jim Hall

On Linux, you may have used a *package manager* to install or remove packages. For example, the default package manager on Debian Linux is the `deb` command, and the default package manager on Fedora Linux is the `dnf` command. But did you know that FreeDOS has a package manager, too?

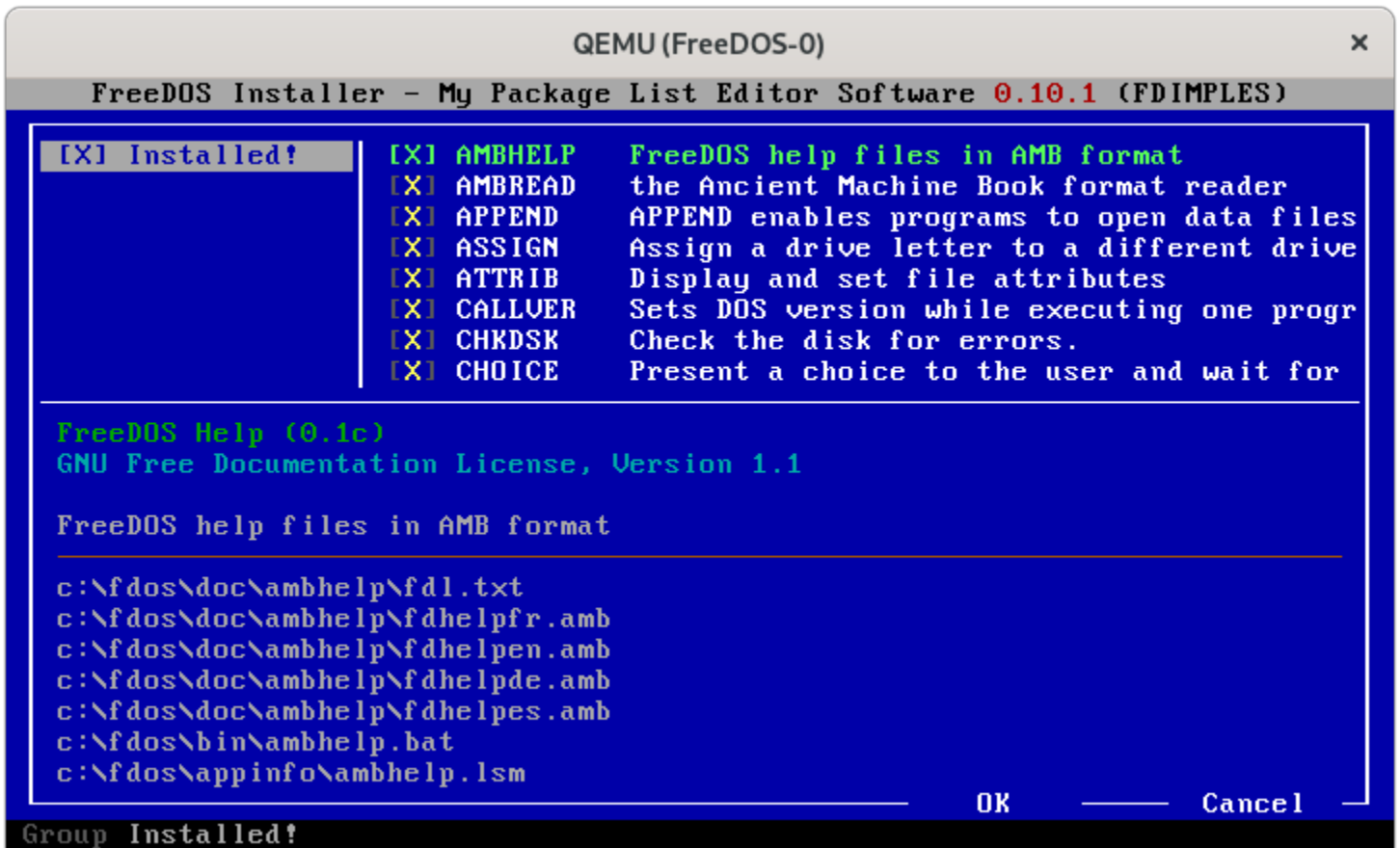
Installing packages

The FreeDOS distribution includes lots of great programs and applications you can use. However, we don't install all of them by default—we don't want to fill your hard drive space unnecessarily, especially on older systems that may have hard drive capacities of only a few hundred megabytes. And if you selected the "Plain DOS system" option when you installed FreeDOS, you will find your FreeDOS system is quite small (about 20 megabytes).

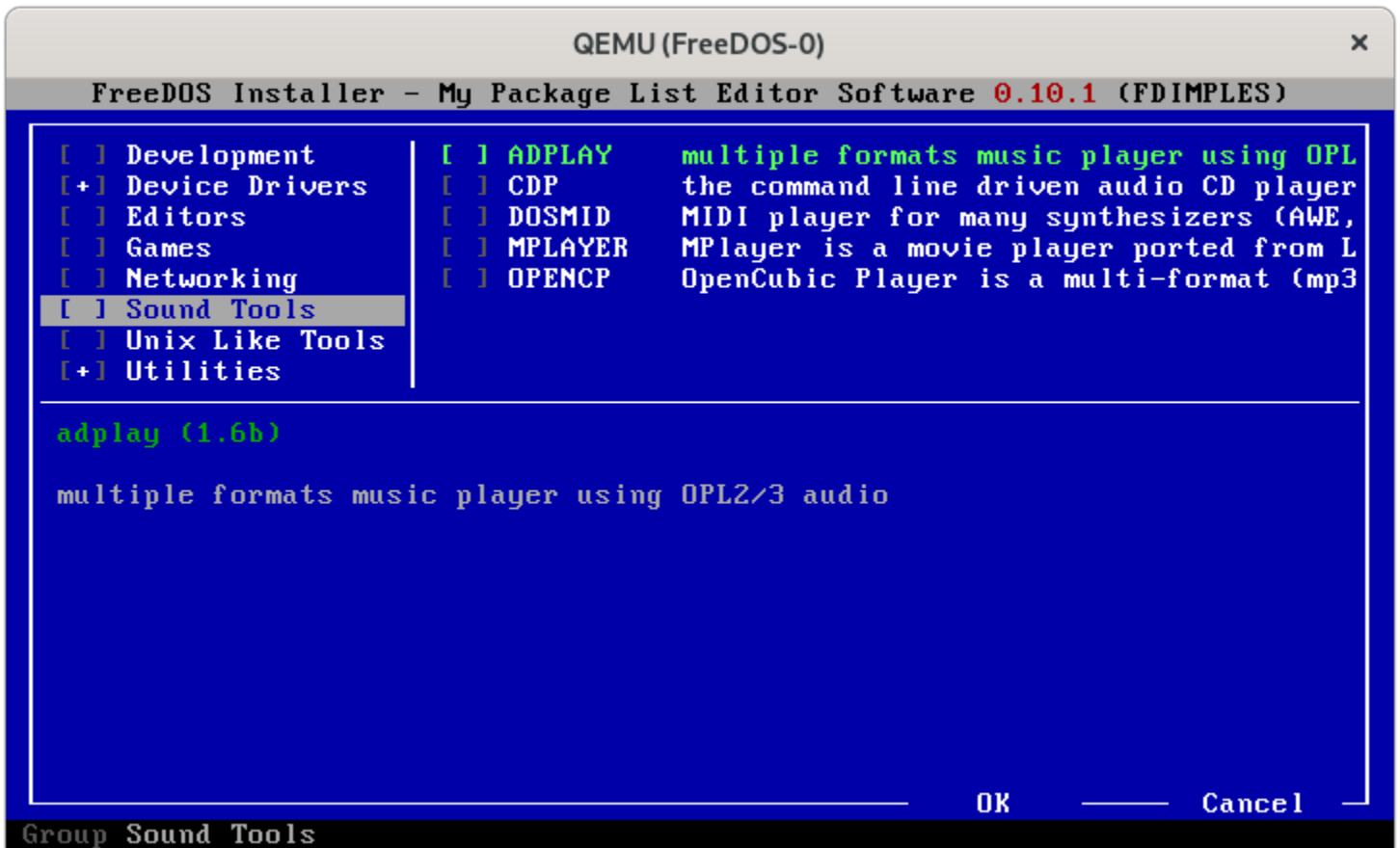
However, it's easy to install new packages. From the command line, run the `FDIMPLES` program. Because DOS is *case insensitive*, you can type this command using uppercase or lowercase letters: `fdimples` is the same as `FDIMPLES` or `FDImples`.



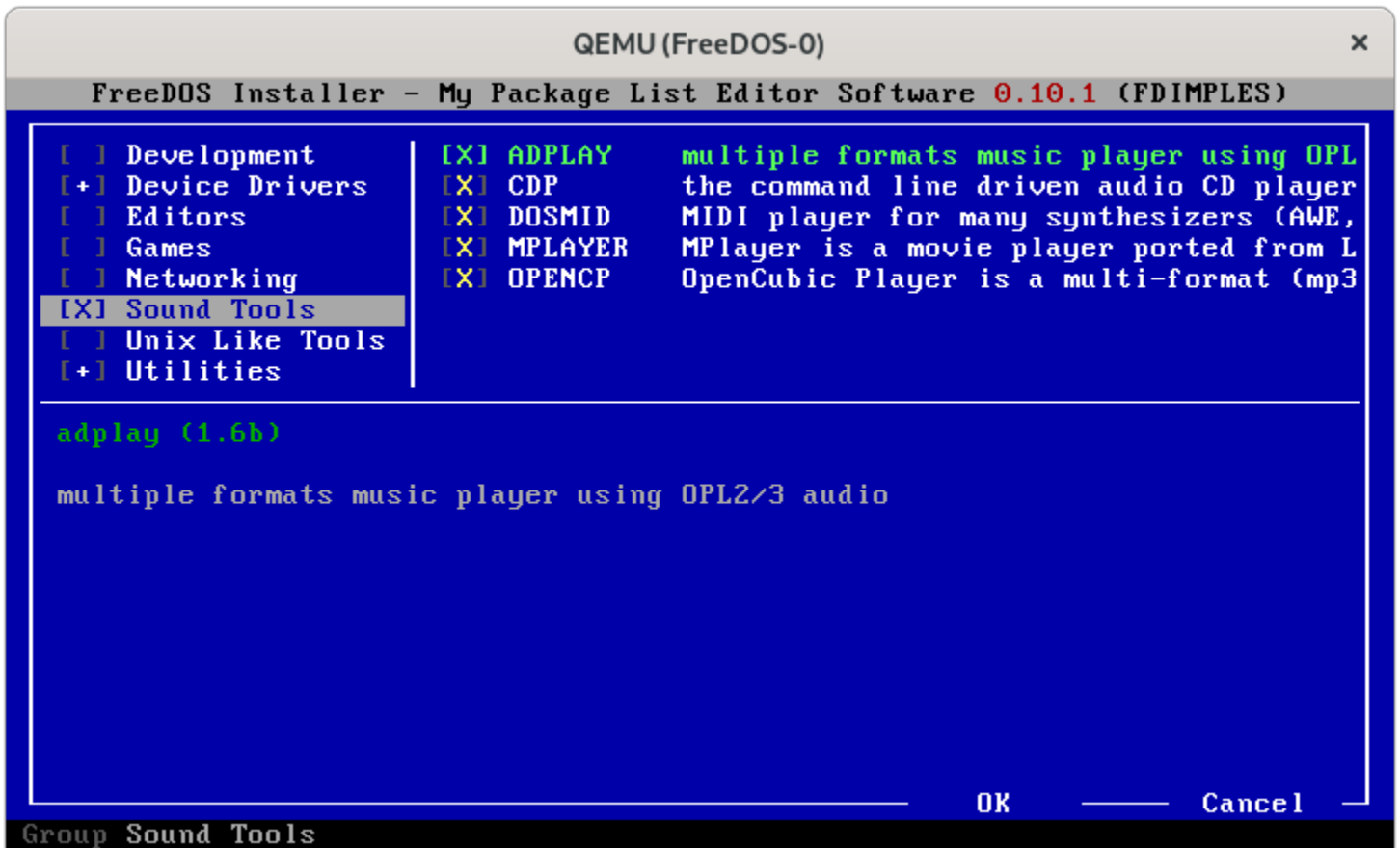
FDIMPLES is an acronym for "FreeDOS Installer - My Package List Editor Software," and is an interactive package manager. FDIMPLES reads the installation media to identify packages that it can install or remove. If you see a different menu that says "Installed" but does not let you install other software, check that the FreeDOS install CD-ROM is loaded on your system. On a physical machine, you'll need to insert the CD-ROM into the CD-ROM drive; on a virtual machine like QEMU or VirtualBox, you'll need to associate the install CD-ROM image with the virtual machine.



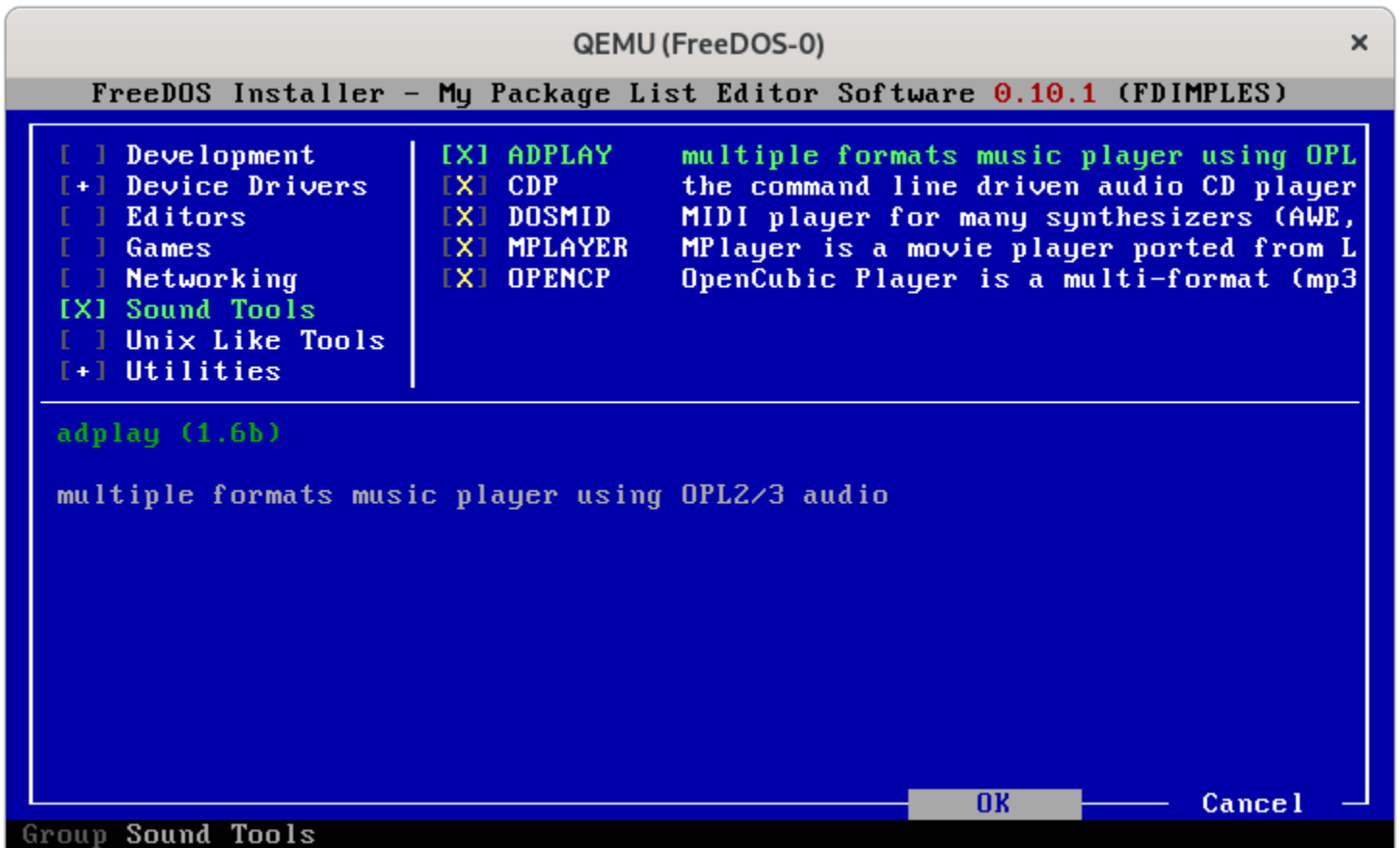
Let's say you wanted to install software that lets you play music and other sound files. Use the Up and Down arrow keys to navigate to the **Sound Tools** entry in the menu. This is called a *package group* for the sound and music programs.



To select all of the packages in this group, press the spacebar on your keyboard. Should you decide to install individual packages in this group, press the Tab key to move into the *package list* pane, then select each package with the spacebar.



You can continue to select other packages or package groups that you want to install. When you have selected everything, use the Tab key to highlight the **OK** button, and press the spacebar.



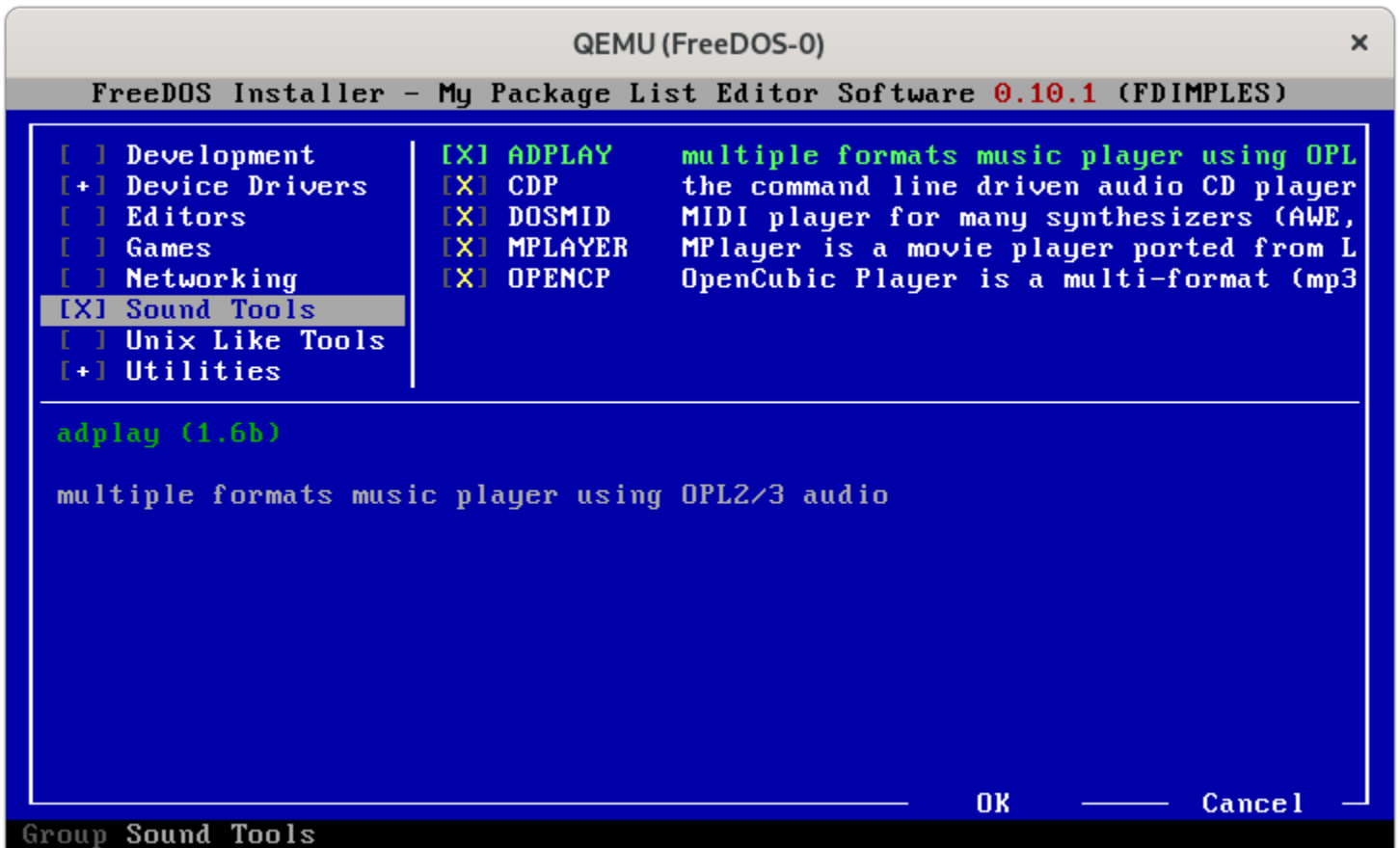
FDIMPLES then installs all of the packages you selected. This may only take a few moments if you selected only a couple of small packages, or it could take minutes if you asked to install many larger packages. As it installs each package, FDIMPLES prints the status. Afterward, FDIMPLES exits to the command line, so you can get back to work.

```
QEMU (FreeDOS-0) x
appinfo\mplayer.de -> C:\FDOS\appinfo\
appinfo\mplayer.fr -> C:\FDOS\appinfo\
appinfo\mplayer.tr -> C:\FDOS\appinfo\
appinfo\mplayer.lsm -> C:\FDOS\appinfo\
Package mplayer installed: 75 files extracted, 0 errors.
install d:\packages\sound\opencp.zip
sound\opencp\ryg.gif -> c:\apps\opencp\
sound\opencp\cp.pak -> c:\apps\opencp\
sound\opencp\readme.txt -> c:\apps\opencp\
sound\opencp\cpghost.exe -> c:\apps\opencp\
sound\opencp\system\apc.oxd -> c:\apps\opencp\system\
sound\opencp\cp.exe -> c:\apps\opencp\
sound\opencp\daposer.gif -> c:\apps\opencp\
sound\opencp\cparcs.dat -> c:\apps\opencp\
sound\opencp\cpmodnfo.dat -> c:\apps\opencp\
sound\opencp\vocp.dll -> c:\apps\opencp\
sound\opencp\cp.ini -> c:\apps\opencp\
sound\opencp\copying.txt -> c:\apps\opencp\
sound\opencp\whatsnew.txt -> c:\apps\opencp\
appinfo\opencp.fr -> C:\FDOS\appinfo\
appinfo\opencp.tr -> C:\FDOS\appinfo\
appinfo\opencp.de -> C:\FDOS\appinfo\
appinfo\opencp.lsm -> C:\FDOS\appinfo\
Package opencp installed: 17 files extracted, 0 errors.
C:\>
```

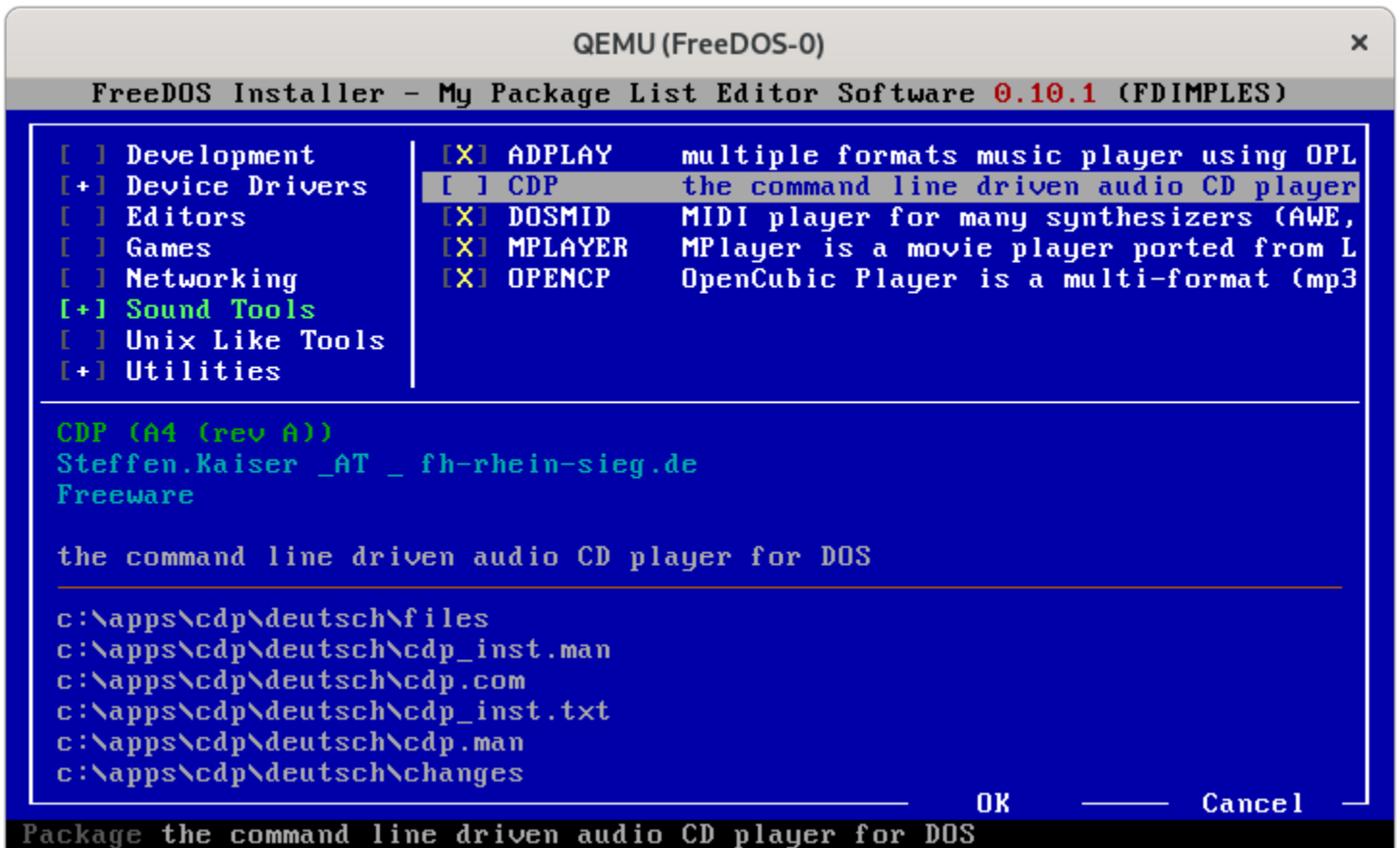
Removing packages

What if you install a package, only to discover afterward that you don't want it? Removing packages is just as easy in FDIMPLES.

Just as when installing packages, start FDIMPLES, and use the arrow keys to navigate to the group that contains the packages you want to remove. For example, if I wanted to uninstall a few of the music-player packages I installed earlier, I would navigate to the **Sound Tools** package group.



To remove the entire package group at once, you can press the spacebar on the group you want to remove to unselect it. But let's say I only wanted to remove a few packages that I didn't need, like the **CDP** audio CD player. (I don't have any physical music CDs.) To remove CDP, hit the Tab key to move into the package list pane, then use the spacebar to unselect the CDP package. This removes the **X** on that package.



You can continue to unselect other packages or package groups that you want to remove. When you have selected everything, use the Tab key to highlight the **OK** button, and press the spacebar. FDIMPLES will remove the packages you unselected, then automatically returns you to the command line.


```
QEMU (FreeDOS-0) x
remove CDP
removing c:\apps\cdp\deutsch\files
removing c:\apps\cdp\deutsch\cdp_inst.man
removing c:\apps\cdp\deutsch\cdp.com
removing c:\apps\cdp\deutsch\cdp_inst.txt
removing c:\apps\cdp\deutsch\cdp.man
removing c:\apps\cdp\deutsch\changes
removing c:\apps\cdp\deutsch\cdp_inst.com
removing c:\apps\cdp\deutsch\cdp.txt
removing c:\apps\cdp\readme.txt
removing c:\apps\cdp\file_id.diz
removing c:\apps\cdp\english\files
removing c:\apps\cdp\english\cdp_inst.man
removing c:\apps\cdp\english\cdp.com
removing c:\apps\cdp\english\cdp_inst.txt
removing c:\apps\cdp\english\cdp.man
removing c:\apps\cdp\english\changes
removing c:\apps\cdp\english\cdp_inst.com
removing c:\apps\cdp\english\cdp.txt
removing C:\FDOS\appinfo\cdp.de
removing C:\FDOS\appinfo\cdp.fr
removing C:\FDOS\appinfo\cdp.tr
removing C:\FDOS\appinfo\cdp.lsm
Package CDP has been removed.
C:\>
```

We include many packages in FreeDOS, across a variety of package groups. Use FDIMPLES to install the software you need. Feel free to experiment! If you decide you don't want to keep a package, you can remove it and free up your disk space for other things.

Install FreeDOS without the installer

By Jim Hall

Most people should be able to install FreeDOS very easily using the installer. The FreeDOS installer asks a few questions, then takes care of the rest—including making space for FreeDOS and making the system bootable.

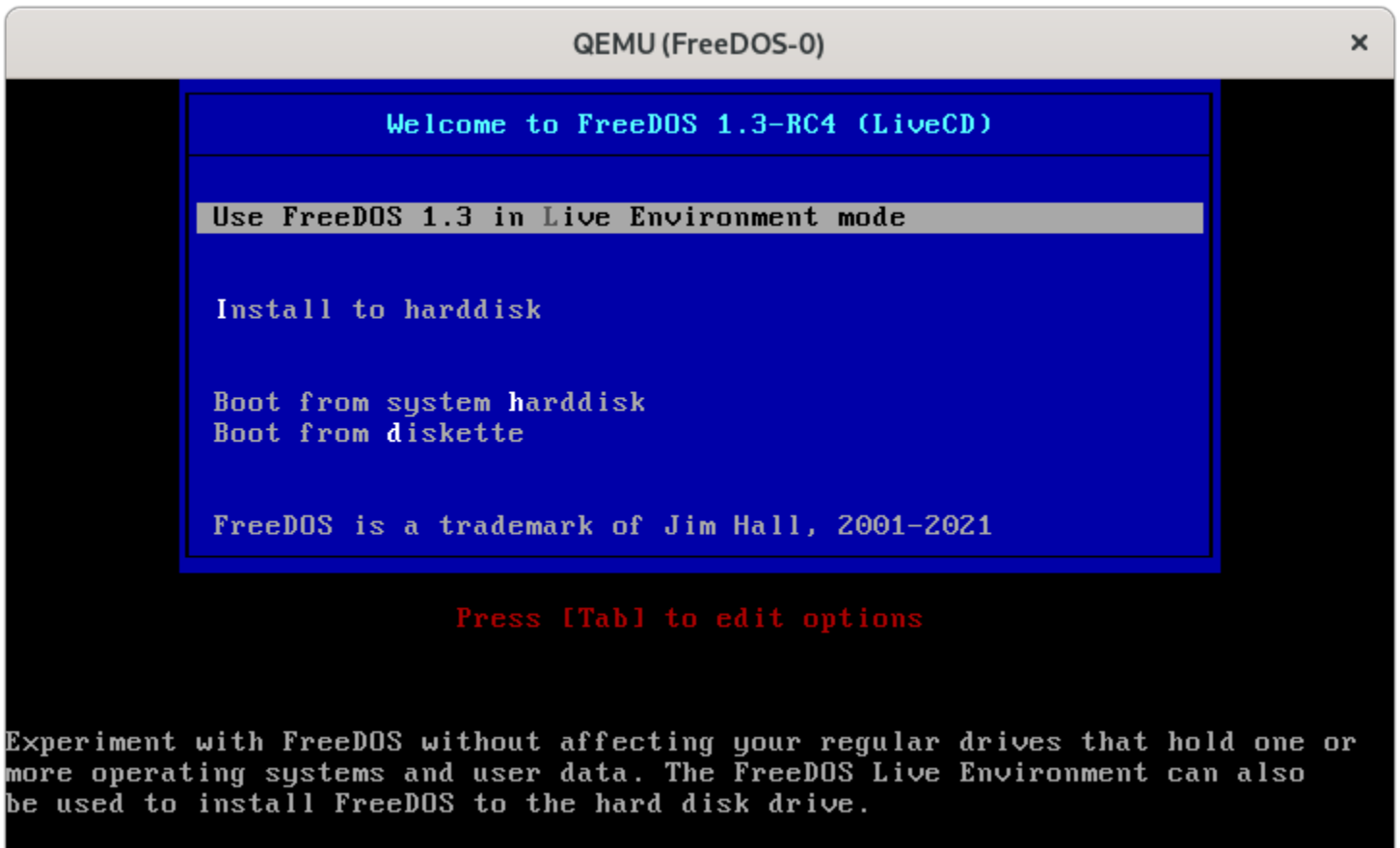
But what if the installer doesn't work for you? Or what if you prefer to set up your FreeDOS system *manually*, without using the installer? With FreeDOS, you can do that too! Let's walk through the steps to install FreeDOS without using the installer. I'll do all of these steps using the QEMU virtual machine, using a blank hard drive image. I created a one hundred megabyte ("100M") hard drive image with this Linux command:

```
$ qemu-img create freedos.img 100M
```

I downloaded the FreeDOS installation LiveCD as FD13LIVE.iso, which provides a "live" environment where I can run FreeDOS, including all the standard tools. Most users also use the LiveCD to install FreeDOS with the regular installer, but here I'll only use the LiveCD to install FreeDOS using individual commands from the command line.

I started the virtual machine with this rather long QEMU command, and selected the "Use FreeDOS in Live Environment mode" boot menu entry:

```
$ qemu-system-x86_64 -name FreeDOS -machine pc-i440fx-4.2,accel=kvm,usb=off,dump-guest-core=off -enable-kvm -cpu host -m 8 -overcommit mem-lock=off -no-user-config -nodefaults -rtc base=utc,driftfix=slew -no-hpet -boot menu=on,strict=on -sandbox on,obsolete=deny,elevateprivileges=deny,spawn=deny,resourcecontrol=deny -msg timestamp=on -hda freedos.img -cdrom FD13LIVE.iso -device sb16 -device adlib -soundhw pcspk -vga cirrus -display sdl -usbdevice mouse
```



Select "Use FreeDOS in Live Environment mode" to boot the LiveCD
(Jim Hall, [CC-BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

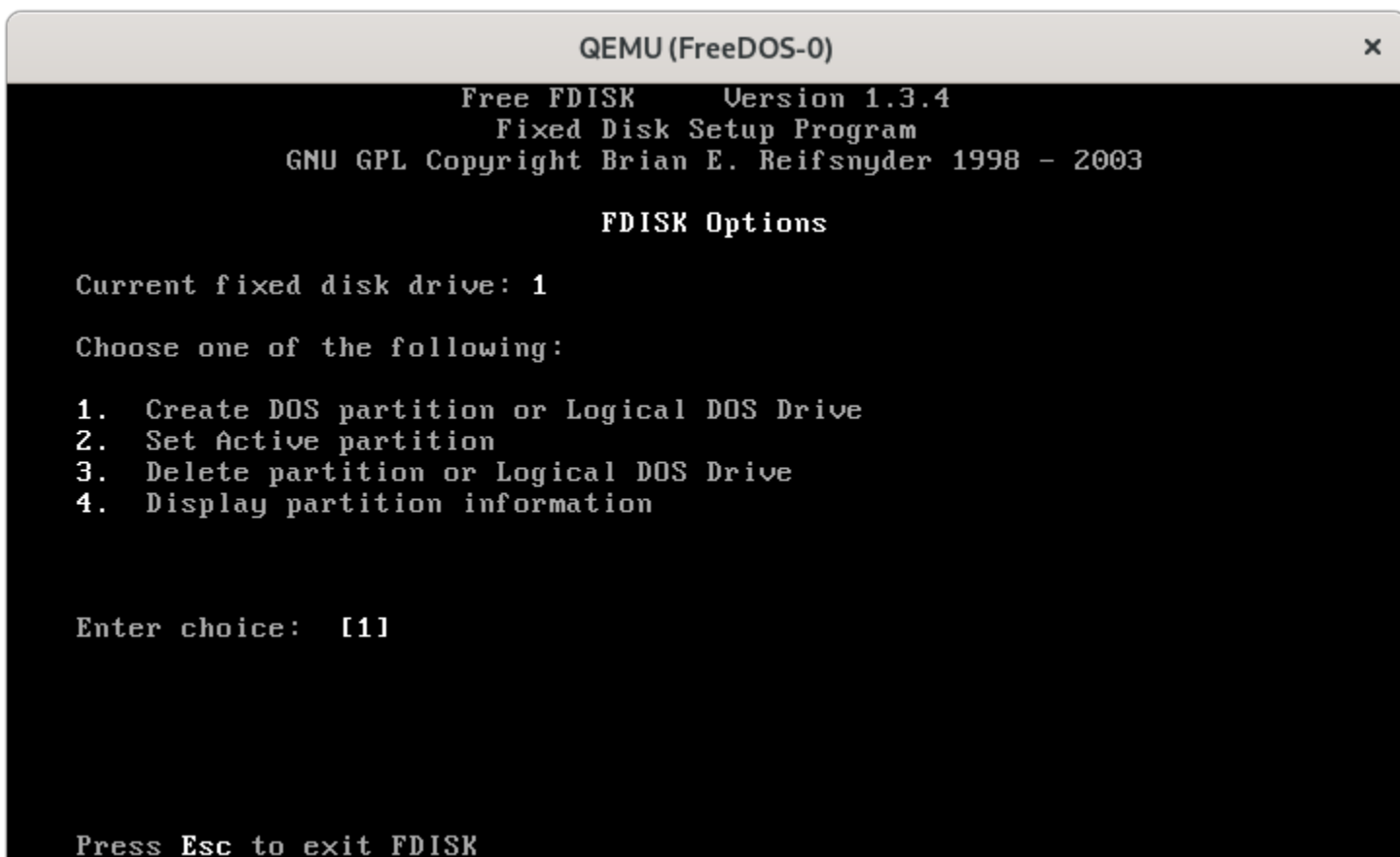
That QEMU command line includes a bunch of options that may seem confusing at first. You configure QEMU entirely with command-line options, so there is a lot to examine here. But I'll briefly highlight a few important options:

- **-m 8**: Set the system memory ("RAM") to 8 megabytes
- **-boot menu=on,strict=on**: Use a boot menu, so I can select if I want to boot from the CD-ROM image or the hard drive image
- **-hda freedos.img**: Use **freedos.img** as the hard drive image
- **-cdrom FD13LIVE.iso**: Use **FD13LIVE.iso** as the CD-ROM image
- **-device sb16 -device adlib -soundhw pcspk**: Define the machine with a SoundBlaster16 sound card, AdLib digital music card, and PC speaker emulation (these are useful if you want to play DOS games)
- **-usbdevice mouse**: Recognize the user's mouse as a USB mouse (click in the QEMU window to use the mouse)

Partition the hard drive

You can use FreeDOS from the LiveCD, but if you want to install FreeDOS to your computer, you'll first need to make space on the hard drive. This requires creating a *partition* with the FDISK program.

From the DOS command line, type `FDISK` to run the *fixed disk* setup program. FDISK is a full-screen interactive program, and you only need to type a number to select a menu item. From the main FDISK menu, enter "1" to create a DOS partition on the drive, then enter "1" on the next screen to create a *primary* DOS partition.

A screenshot of a QEMU window titled "QEMU (FreeDOS-0)". The window contains a text-based interface for the FDISK program. The text is as follows:

```
Free FDISK      Version 1.3.4
Fixed Disk Setup Program
GNU GPL Copyright Brian E. Reifsnyder 1998 - 2003

FDISK Options

Current fixed disk drive: 1

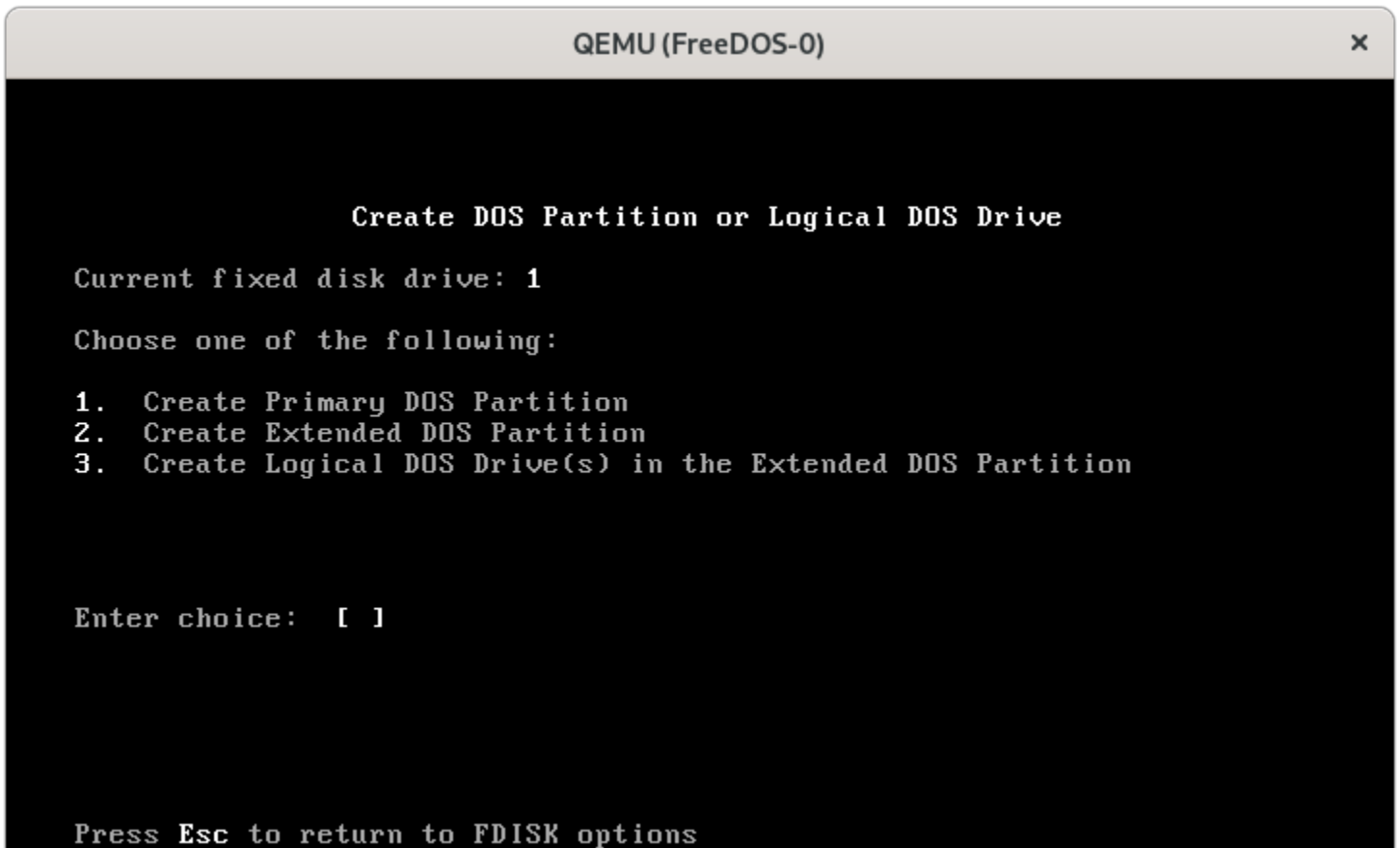
Choose one of the following:

1. Create DOS partition or Logical DOS Drive
2. Set Active partition
3. Delete partition or Logical DOS Drive
4. Display partition information

Enter choice: 11

Press Esc to exit FDISK
```

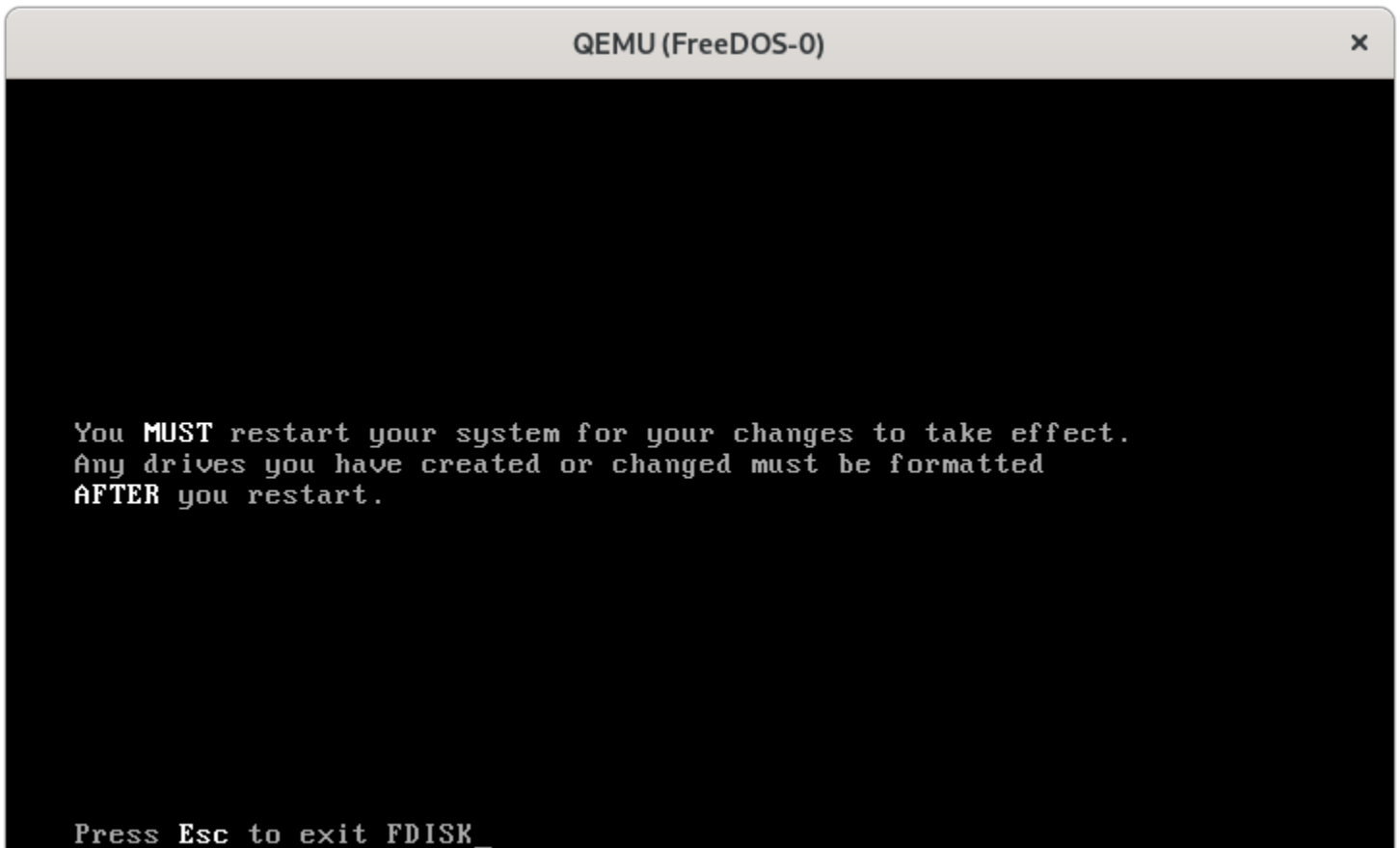
Select "1" to create a partition
(Jim Hall, [CC-BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))



Select "1" on the next menu to make a primary partition
(Jim Hall, [CC-BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

FDISK asks you if you wish to use the full size of the hard disk to create the partition. Unless you need to share space on this hard drive with another operating system, such as Linux, you should answer "Y" to this prompt.

After FDISK creates the new partition, you'll need to reboot before DOS can recognize the new partition information. Like all DOS operating systems, FreeDOS identifies the hard drive information only when it boots up. So if you create or delete any disk partitions, you'll need to reboot so FreeDOS recognizes the changed partition information. FDISK reminds you to reboot, so you don't forget.



You need to reboot to recognize the new partition
(Jim Hall, [CC-BY SA 4.0](#))

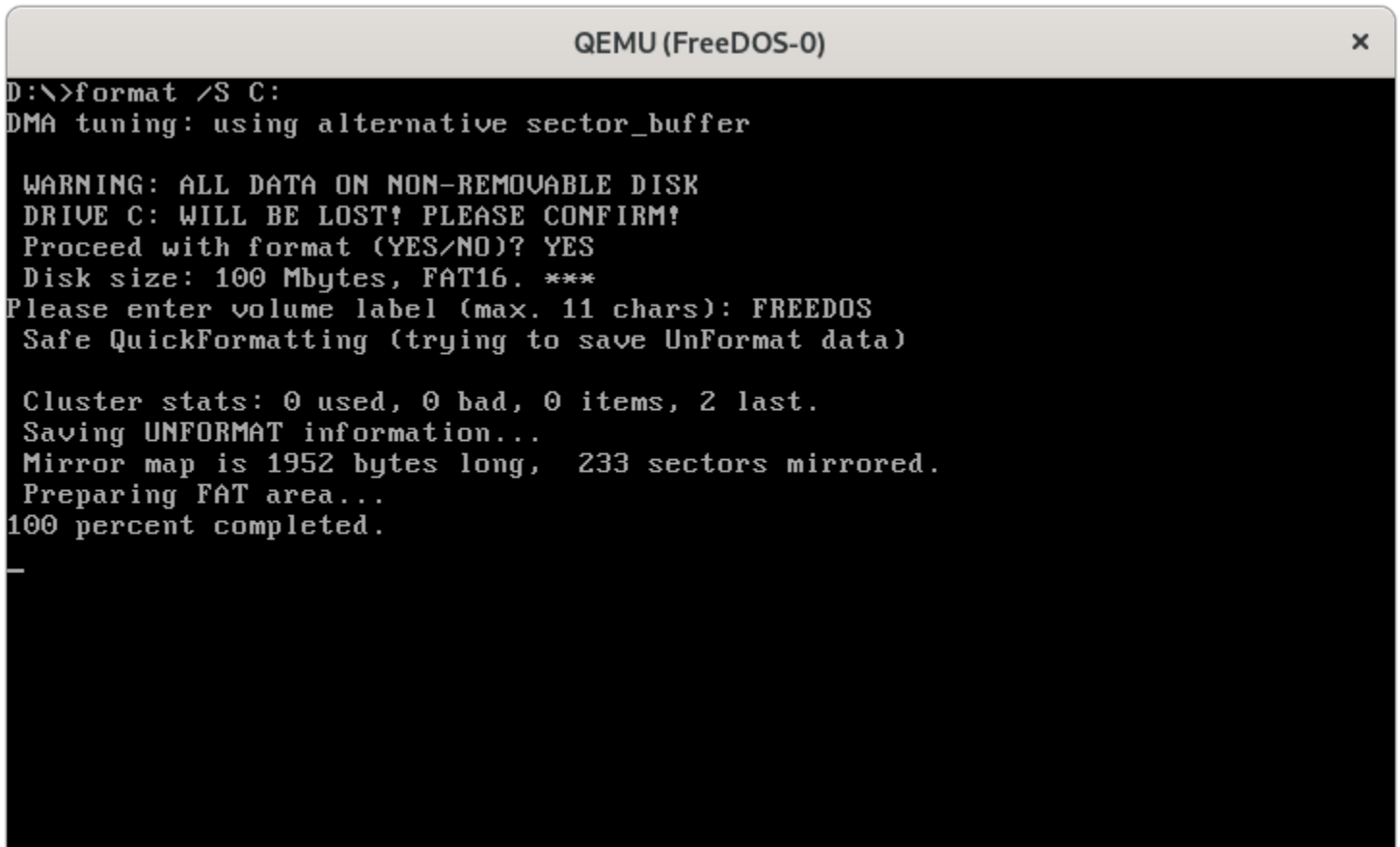
You can reboot by stopping and restarting the QEMU virtual machine, but I prefer to reboot FreeDOS from the FreeDOS command line, using the FreeDOS Advanced Power Management (FDADPM) tool. To reboot, type the command `FDADPM /WARMBOOT` and FreeDOS reboots itself.

Formatting the hard drive

After FreeDOS restarts, you can continue setting up the hard drive. Creating the disk partition was "step 1" in this process; now you need to make a DOS *filesystem* on the partition so FreeDOS can use it.

DOS systems identify "drives" using the letters A through Z. FreeDOS recognizes the first partition on the first hard drive as the C drive, and so on. You often indicate the drive with the letter and a colon (:) so the new partition we created above is actually the C: drive.

You can create a DOS filesystem on the new partition with the FORMAT command. This command takes a few options, but we'll only use the /S option to tell FORMAT to make the new filesystem bootable—the "S" means to install the FreeDOS "System" files. Type `FORMAT /S C:` to make a new DOS filesystem on the C: drive.



```
QEMU (FreeDOS-0) x
D:\>format /S C:
DMA tuning: using alternative sector_buffer

WARNING: ALL DATA ON NON-REMOVABLE DISK
DRIVE C: WILL BE LOST! PLEASE CONFIRM!
Proceed with format (YES/NO)? YES
Disk size: 100 Mbytes, FAT16. ***
Please enter volume label (max. 11 chars): FREEDOS
Safe QuickFormatting (trying to save UnFormat data)

Cluster stats: 0 used, 0 bad, 0 items, 2 last.
Saving UNFORMAT information...
Mirror map is 1952 bytes long, 233 sectors mirrored.
Preparing FAT area...
100 percent completed.
```

Format the partition to create the DOS filesystem
(Jim Hall, [CC-BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

With the /S option, FORMAT will run the SYS program to transfer the system files. You'll see this as part of the output from FORMAT:

```
QEMU (FreeDOS-0) x
Safe QuickFormat complete.

Running SYS in a shell: sys C:
FreeDOS System Installer v3.6e, May 11 2016

Processing boot sector...
FAT type: FAT16
Now copying system files...
Copying D:KERNEL.SYS...
46685 Bytes transferred
Copying shell (command interpreter)...
Copying D:COMMAND.COM...
85048 Bytes transferred

System transferred.

    104,735,232 bytes total disk space (disk size)
    104,513,536 bytes available on disk (free clusters)

        2,048 bytes in each allocation unit.
    51,032 allocation units on disk.

Volume Serial Number is 2A63-18FA
D:\>
```

FORMAT /S will use SYS to make the disk bootable
(Jim Hall, [CC-BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

Installing software

Having created a new partition with FDISK and a new filesystem with FORMAT, the new C: drive is basically empty. At this point, the C: drive only contains a copy of the kernel and the COMMAND.COM command-line shell. To do anything useful with the new disk, we need to install software on it. This is the last step for the manual install process.

The FreeDOS LiveCD contains all of the software you might want to install on the new system. Each FreeDOS program is available as a separate "package," which is really just a Zip archive file. The packages that set up the standard DOS environment are stored in the BASE directory, under the PACKAGES directory on the LiveCD.

You could install the packages by "unzipping" each of them to the hard drive, one at a time. With 62 individual packages in the "Base" group, installing each package individually

would take a very long time. However, you can run a one-line **FOR** "loop" command to "unzip" each program. Then FreeDOS can "unzip" all of the packages for you.

The basic usage of the **FOR** loop indicates a single-letter variable (let's use %F) that FreeDOS uses to "fill in" the filename later. The **FOR** also requires a list of files in brackets and the command that it should run against each of the files. The syntax to unzip a list of Zip files looks like this:

```
FOR %F IN (*.ZIP) DO UNZIP %F
```

This extracts all of the Zip files into the current directory. To extract or "unzip" the files into a different location, use the **-d** ("destination") option at the end of the **UNZIP** command line. For most FreeDOS systems, you will want to install the software packages to the **C:\FDOS** directory:

```
QEMU (FreeDOS-0) x
D:\PACKAGES\BASE>dir /w
Volume in drive D is FD13-LiveCD

Directory of D:\PACKAGES\BASE

[.]                [..]              AMBHELP.ZIP       APPEND.ZIP        ASSIGN.ZIP
ATTRIB.ZIP         CHKDSK.ZIP        CHOICE.ZIP        COMP.ZIP          CPIDOS.ZIP
CTMOUSE.ZIP        DEBUG.ZIP         DEFRAG.ZIP        DELTREE.ZIP      DEVLOAD.ZIP
DISKCOMP.ZIP       DISKCOPY.ZIP      DISPLAY.ZIP       EDIT.ZIP          EDLIN.ZIP
EXE2BIN.ZIP        FC.ZIP            FDAPM.ZIP         FDHELPER.ZIP     FDISK.ZIP
FDXMS.ZIP          FDXMS286.ZIP     FIND.ZIP          FORMAT.ZIP        FREECOM.ZIP
GRAPHICS.ZIP       HIMEMX.ZIP        HTMLHELP.ZIP     INDEX.DE          INDEX.FR
INDEX.GZ           INDEX.LST         INDEX.TR          JEMM.ZIP         KERNEL.ZIP
KEYB.ZIP           KEYB_LAY.ZIP     LABEL.ZIP         LBACACHE.ZIP     MEM.ZIP
MIRROR.ZIP         MKEYB.ZIP        MODE.ZIP          MORE.ZIP          MOVE.ZIP
NANSI.ZIP          NLSFUNC.ZIP      PRINT.ZIP         RECOVER.ZIP      REPLACE.ZIP
SAMCFG.ZIP         SHARE.ZIP         SHSUCDX.ZIP      SORT.ZIP          SWSUBST.ZIP
TREE.ZIP           UNDELETE.ZIP     UNFORMAT.ZIP     XCOPY.ZIP        62 file(s)
11,619,461 bytes
2 dir(s)           0 bytes free
D:\PACKAGES\BASE>for %f in (*.zip) do unzip %f -d C:\FDOS_
```

Unzip all of the Base packages to finish installing FreeDOS
(Jim Hall, [CC-BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

FreeDOS takes care of the rest, installing all 62 packages to your system. This may take several minutes because DOS can be slow when working with lots of individual files—and this command needs to extract 62 Zip files. The installation process would run a lot faster if we used a single `BASE.ZIP` archive file, but using packages provides more flexibility in what software you might want to install versus what you choose to leave out.

```
QEMU (FreeDOS-0) x
inflating: C:/FDOS/BIN/XCOPY.EXE
creating: C:/FDOS/DOC/RXCOPY/
inflating: C:/FDOS/DOC/RXCOPY/COPYING.TXT
inflating: C:/FDOS/DOC/RXCOPY/XCOPY.TXT
inflating: C:/FDOS/DOC/RXCOPY/HISTORY.TXT
inflating: C:/FDOS/NLS/XCOPY.SL
inflating: C:/FDOS/NLS/XCOPY.PL
inflating: C:/FDOS/NLS/XCOPY.RU
inflating: C:/FDOS/NLS/XCOPY.IT
inflating: C:/FDOS/NLS/XCOPY.DE
inflating: C:/FDOS/NLS/XCOPY.TR
inflating: C:/FDOS/NLS/XCOPY.FR
inflating: C:/FDOS/NLS/XCOPY.ES
inflating: C:/FDOS/NLS/XCOPY.EN
creating: C:/FDOS/SOURCE/XCOPY/
inflating: C:/FDOS/SOURCE/XCOPY/TCPP3.MAK
inflating: C:/FDOS/SOURCE/XCOPY/XCOPY.MAP
inflating: C:/FDOS/SOURCE/XCOPY/PRF.C
inflating: C:/FDOS/SOURCE/XCOPY/MAKEFILE
inflating: C:/FDOS/SOURCE/XCOPY/SHARED.INC
inflating: C:/FDOS/SOURCE/XCOPY/TURBOC.CFG
inflating: C:/FDOS/SOURCE/XCOPY/KITTEN.H
inflating: C:/FDOS/SOURCE/XCOPY/XCOPY.C
inflating: C:/FDOS/SOURCE/XCOPY/KITTEN.C
D:\PACKAGES\BASE>
```

After installing all the Base packages

(Jim Hall, [CC-BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

After you've installed everything, reboot your system with `FDADPM /WARMBOOT`. Installing manually means your new FreeDOS system won't have the usual `FDCONFIG.SYS` configuration file, so FreeDOS will assume a few typical default values when it starts up. Without the `AUTOEXEC.BAT` file, FreeDOS also prompts you for the time and date.

```
QEMU (FreeDOS-0) x
SeaBIOS (version 1.14.0-2.fc34)

Press ESC for boot menu.

Booting from Hard Disk...
FreeDOS kernel 2042 (build 2042 DEM:0xfd) [compiled May 11 2016]
Kernel compatibility 7.10 - WATCOMC - FAT32 support

(C) Copyright 1995-2012 Pasquale J. Uillani and The FreeDOS Project.
All Rights Reserved. This is free software and comes with ABSOLUTELY NO
WARRANTY; you can redistribute it and/or modify it under the terms of the
GNU General Public License as published by the Free Software Foundation;
either version 2, or (at your option) any later version.
C: HD1, Pri[ 1], CHS=    0-1-1, start=    0 MB, size=    99 MB

FreeCom version 0.84-pre7 - WATCOMC - XMS_Swap [Dec 29 2019 15:32:39]
Current date is Fri 05-14-2021
Enter new date (mm-dd-[cc]yy):
Current time is 5:33:33.20 pm
Enter new time:
C:\>_
```

Rebooting FreeDOS after a manual install
(Jim Hall, [CC-BY SA 4.0](#))

Most users should be able to use the more user-friendly process to install FreeDOS on a new computer. But if you want to install it yourself the "old school" way, you can also run the installation steps manually. This can provide some additional flexibility and control because you install everything yourself. And now you know how to do it.

How to use FreeDOS as an embedded system

By Jim Hall

The [FreeDOS website](#) says that most people use FreeDOS for three main tasks:

1. Playing classic DOS games
2. Running legacy DOS software
3. Running an embedded system

But what does it mean to run an "embedded" system?

An embedded system is basically a very minimal system that is dedicated to run a specific task. You might think of embedded systems today as part of the *Internet of Things* (IoT) including sensors, thermostats, and doorbell cameras. Many embedded systems today run on Linux.

But once upon a time, embedded systems either ran on a custom, proprietary platform or ran on DOS. Some of these DOS-based embedded systems still run today, such as cash registers or phone private branch exchange (PBX) systems. In one example as recently as 2017, trainspotters discovered a Russian electric train control system (Russian: *САВЛТЭ*) running FreeDOS with special software to control and monitor the route of suburban trains and to make passenger announcements.

Setting up an embedded system on DOS requires defining a minimal DOS environment that runs a single application. Fortunately, setting up a minimal FreeDOS environment is pretty easy. Technically, all you need to boot FreeDOS and run DOS applications is the kernel and a `FDCONFIG.SYS` configuration file.

Installing a minimal system

We can simulate a dedicated, minimal FreeDOS system by using the QEMU emulator with very small allocations. To reflect an embedded system more accurately, I'll define a virtual machine with only 8 megabytes of memory and a mere 2 megabytes for a virtual hard drive.

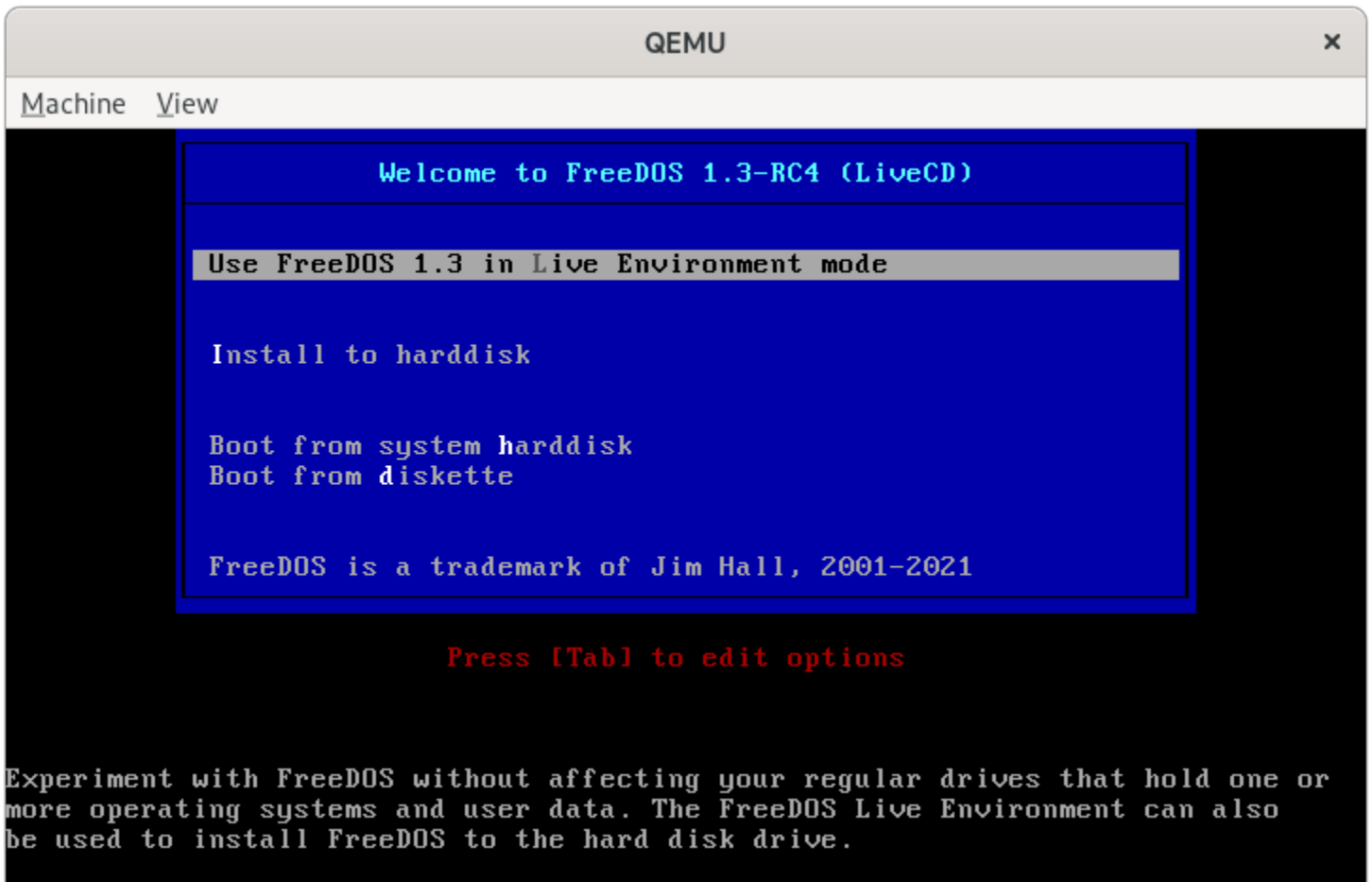
To create the tiny virtual hard drive, I'll use this `qemu-img` command to define a 2-megabyte file:

```
$ qemu-img create tiny.img 2M
Formatting 'tiny.img', fmt=raw size=2097152
```

This command line defines a 32-bit "i386" CPU with 8 megabytes of memory, using the 2-megabyte `tiny.img` file as the hard drive image and the FreeDOS LiveCD as the CD-ROM media. We'll also set the machine to boot from the CD-ROM drive (`-boot order=d`) although we only need that to install. We'll boot the completed embedded system from the hard disk after we've set everything up:

```
qemu-system-i386 -m 8 -hda tiny.img -cdrom FD13LIVE.iso -boot order=d
```

Boot the system using the "Live Environment mode"—this provides us with a running FreeDOS system that we can use to transfer a minimal FreeDOS to the hard disk.



Boot into the LiveCD environment
(Jim Hall, [CC-BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

We'll need to create a partition on the virtual hard drive for our programs. To do that, run the FDISK program from the command line. FDISK is the standard *fixed disk* utility on FreeDOS. Use FDISK to create a single hard drive partition that spans the entire (2-megabyte) hard drive.

```
QEMU x
Machine View

                Create Primary DOS Partition

Current fixed disk drive: 1

Partition  Status  Type      Volume Label  Mbytes  System  Usage
C: 1      A          PRI DOS                    2      FAT12   100%

Total disk space is      2 Mbytes (1 Mbyte = 1048576 bytes)

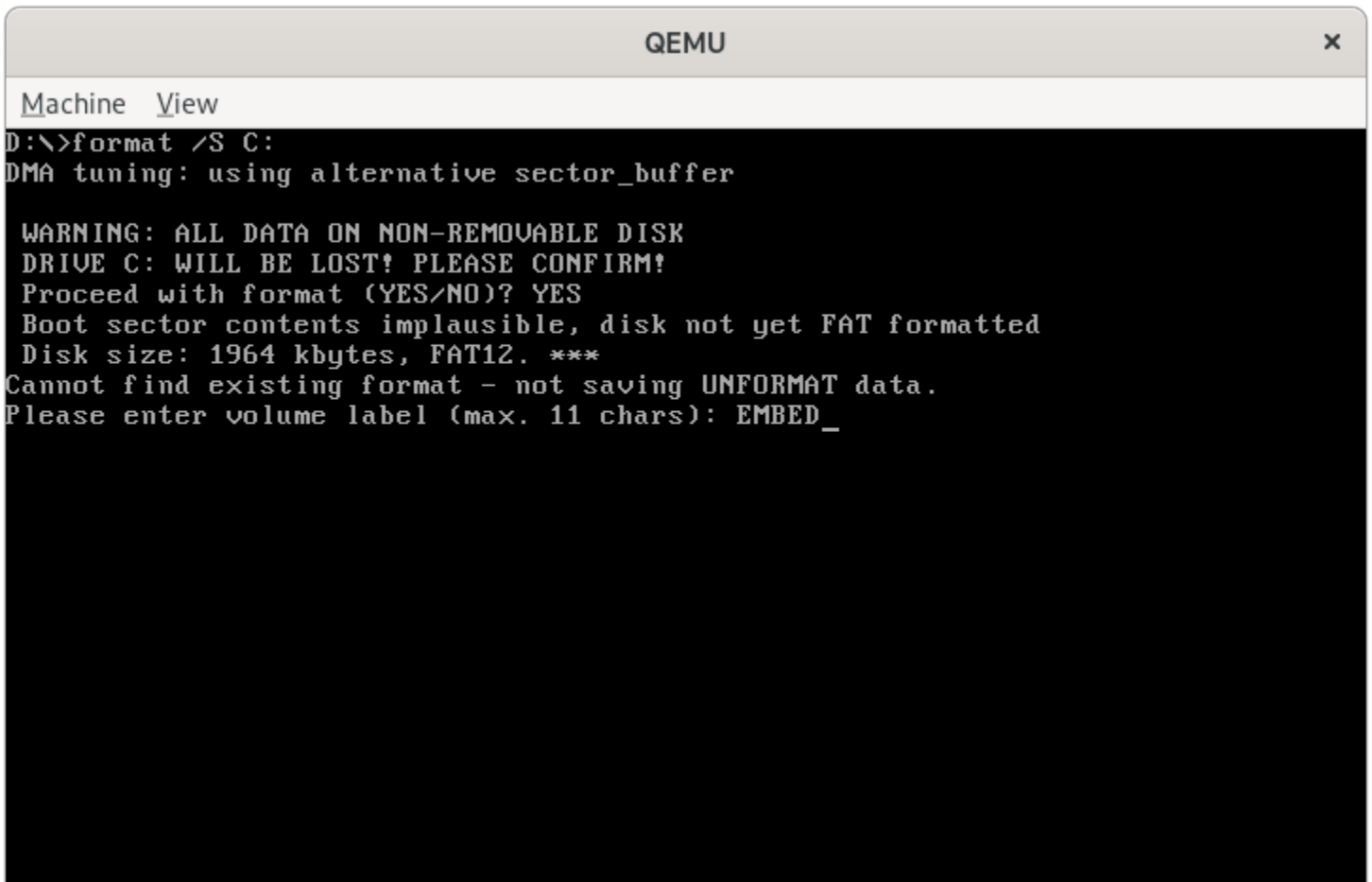
Primary DOS Partition created

Press Esc to continue
```

FDISK, after creating the 2 megabyte partition
(Jim Hall, [CC-BY SA 4.0](#))

But FreeDOS won't see the new hard drive partition until you reboot—FreeDOS only reads the hard disk details at startup. Exit FDISK and reboot, and you'll be ready for the next step.

After rebooting, you need to create a DOS filesystem on the new hard drive. Since there's just the one virtual hard disk, FreeDOS will identify it as the C: drive. You can create a DOS filesystem on C: with the FORMAT command. The /S option transfers the operating system files (the kernel, plus a copy of the COMMAND.COM shell) to the new drive.

A screenshot of a QEMU terminal window. The window title is "QEMU" with a close button "x" in the top right corner. Below the title bar is a menu bar with "Machine" and "View". The terminal content shows a DOS command prompt where the user has entered "format /S C:". The output includes "DMA tuning: using alternative sector_buffer", a warning about data loss on a non-removable disk, a confirmation prompt "Proceed with format (YES/NO)? YES", a message "Boot sector contents implausible, disk not yet FAT formatted", "Disk size: 1964 kbytes, FAT12. ***", "Cannot find existing format - not saving UNFORMAT data.", and a prompt for a volume label "Please enter volume label (max. 11 chars): EMBED_".

```
Machine View
D:\>format /S C:
DMA tuning: using alternative sector_buffer

WARNING: ALL DATA ON NON-REMOVABLE DISK
DRIVE C: WILL BE LOST! PLEASE CONFIRM!
Proceed with format (YES/NO)? YES
Boot sector contents implausible, disk not yet FAT formatted
Disk size: 1964 kbytes, FAT12. ***
Cannot find existing format - not saving UNFORMAT data.
Please enter volume label (max. 11 chars): EMBED_
```

Format the new drive to create a DOS filesystem
(Jim Hall, [CC-BY SA 4.0](#))

Now that you've created the drive and formatted it, you can install the application that will run on the embedded system.

Installing the dedicated application

An embedded system is really just a single-purpose application running on a dedicated system. Such applications are usually custom-built for the system it will control, such as a cash register, display terminal, or control environment. For this demonstration, let's use a program from the FreeDOS installation CD-ROM. It needs to be small enough to fit in the tiny 2-megabyte hard drive we've created for it. This can be anything—so just for fun, let's make it a game.

FreeDOS includes several fun games. One game that I like is a board game called Simple Senet. It's based on Senet, an ancient Egyptian board game. The details of the game aren't

important for this demonstration, except that we'll install it and set it up as the dedicated application for the embedded system.

To install the application, go into the \PACKAGES\GAMES directory on the FreeDOS LiveCD. You'll see a long list of packages there, and the one we want is SENET.ZIP.

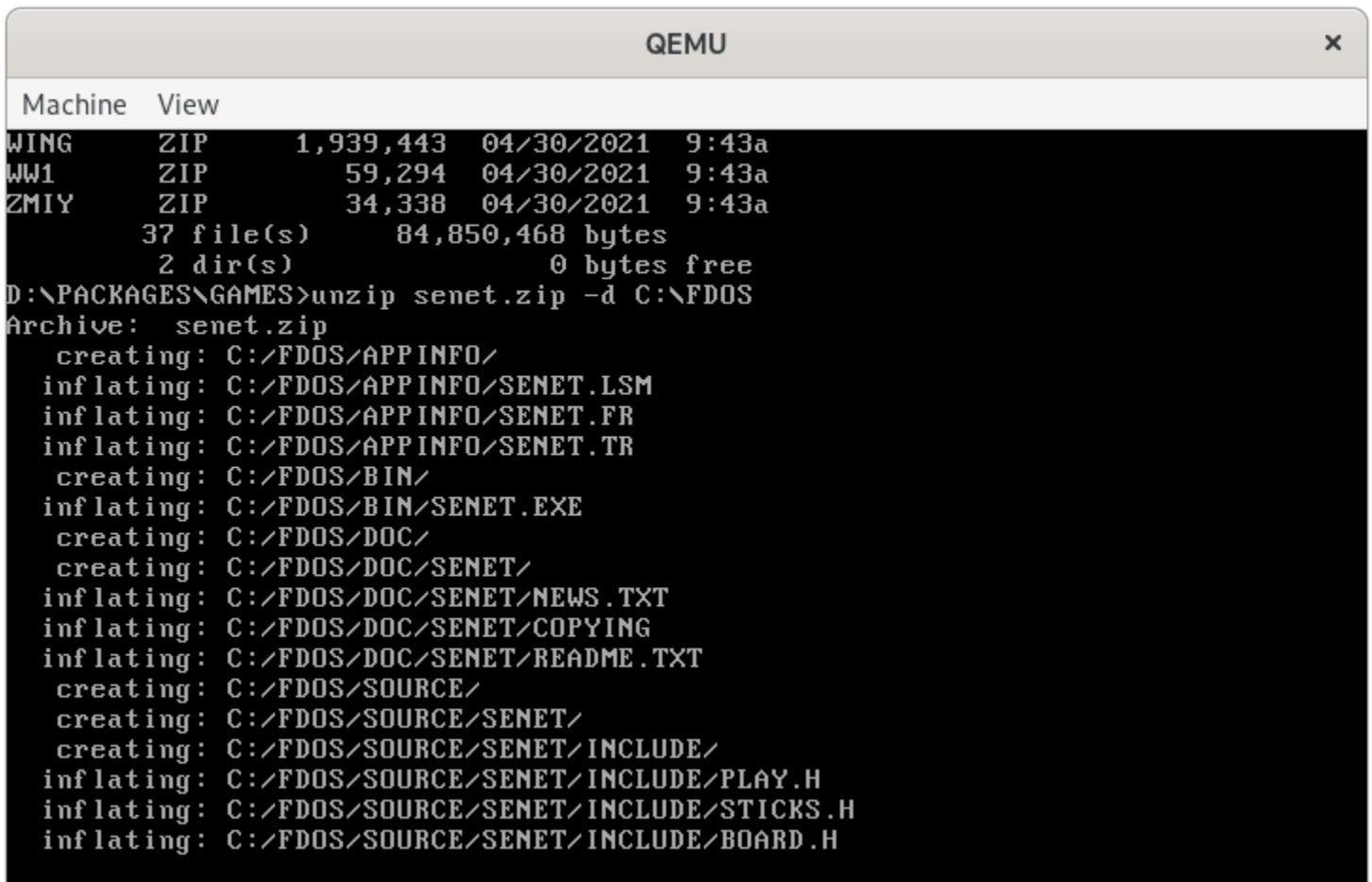
```
QEMU x
Machine View
INDEX GZ 1,803 04/30/2021 9:43a
INDEX LST 3,185 04/30/2021 9:43a
INDEX TR 3,293 04/30/2021 9:43a
IVAN ZIP 2,143,728 04/30/2021 9:43a
KRAPTOR ZIP 10,398,624 04/30/2021 9:43a
LIQUIWAR ZIP 6,818,959 04/30/2021 9:43a
MAGNETIC ZIP 715,017 04/30/2021 9:43a
MIRMAGIC ZIP 1,436,512 04/30/2021 9:43a
MISTRAL ZIP 8,526,598 04/30/2021 9:43a
NETHACK ZIP 4,989,998 04/30/2021 9:43a
NGE_NIBB ZIP 183,116 04/30/2021 9:43a
NOUDAR ZIP 4,904,893 04/30/2021 9:43a
PAKUPAKU ZIP 72,804 04/30/2021 9:43a
QTETRIS ZIP 2,889,210 04/30/2021 9:43a
SENET ZIP 60,107 04/30/2021 9:43a
SNOVA ZIP 779,944 04/30/2021 9:43a
SUDOKU86 ZIP 243,865 04/30/2021 9:43a
VERTIGO ZIP 3,404,921 04/30/2021 9:43a
VITETRIS ZIP 355,616 04/30/2021 9:43a
WING ZIP 1,939,443 04/30/2021 9:43a
WW1 ZIP 59,294 04/30/2021 9:43a
ZMIY ZIP 34,338 04/30/2021 9:43a
37 file(s) 84,850,468 bytes
2 dir(s) 0 bytes free
D:\PACKAGES\GAMES>
```

A list of game packages from FreeDOS
(Jim Hall, [CC-BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

To unzip the Simple Senet package onto the virtual hard drive, use the UNZIP command. All FreeDOS packages are Zip files, so you can use any Zip-compatible archive utility to manage them. FreeDOS includes ZIP to create Zip archives, and UNZIP to extract Zip archives. Both are from the [Info-Zip Project](https://www.info-zip.org/).

```
UNZIP SENET.ZIP -d C:\FDOS
```

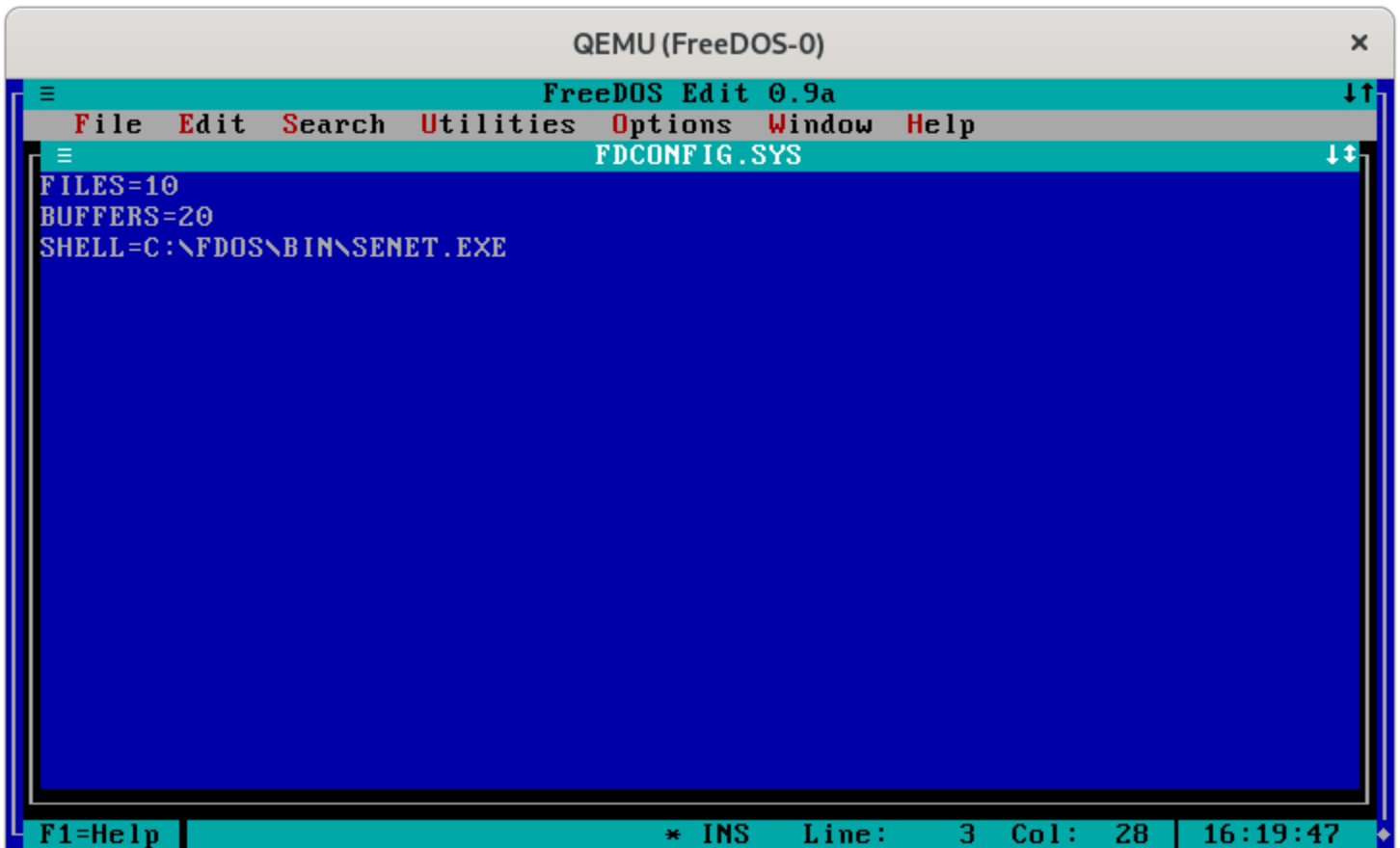
Normally, using UNZIP will extract a Zip file in the current directory. The `-d C:\FDOS` option at the end of the command line tells UNZIP to extract the Zip file to the `C:\FDOS` directory. (`-d` means "destination.")



```
QEMU x
Machine  View
WING     ZIP      1,939,443  04/30/2021  9:43a
WW1      ZIP      59,294     04/30/2021  9:43a
ZMIY     ZIP      34,338     04/30/2021  9:43a
          37 file(s)   84,850,468 bytes
          2 dir(s)    0 bytes free
D:\PACKAGES\GAMES>unzip senet.zip -d C:\FDOS
Archive:  senet.zip
  creating: C:/FDOS/APPINFO/
  inflating: C:/FDOS/APPINFO/SENET.LSM
  inflating: C:/FDOS/APPINFO/SENET.FR
  inflating: C:/FDOS/APPINFO/SENET.TR
  creating: C:/FDOS/BIN/
  inflating: C:/FDOS/BIN/SENET.EXE
  creating: C:/FDOS/DOC/
  creating: C:/FDOS/DOC/SENET/
  inflating: C:/FDOS/DOC/SENET/NEWS.TXT
  inflating: C:/FDOS/DOC/SENET/COPYING
  inflating: C:/FDOS/DOC/SENET/README.TXT
  creating: C:/FDOS/SOURCE/
  creating: C:/FDOS/SOURCE/SENET/
  creating: C:/FDOS/SOURCE/SENET/INCLUDE/
  inflating: C:/FDOS/SOURCE/SENET/INCLUDE/PLAY.H
  inflating: C:/FDOS/SOURCE/SENET/INCLUDE/STICKS.H
  inflating: C:/FDOS/SOURCE/SENET/INCLUDE/BOARD.H
```

Unzipping the Simple Senet game
(Jim Hall, [CC-BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

To run the Simple Senet game whenever the embedded system boots, we need to tell FreeDOS to use Senet as the system "shell." The default FreeDOS shell is the `COMMAND.COM` program, but you can define a different shell program using the `SHELL=` directive in the `FDCONFIG.SYS` kernel configuration file. We can use FreeDOS Edit to create the new `C:\FDCONFIG.SYS` file.



(Jim Hall, [CC-BY SA 4.0](#))

If you need to define other parameters to support the embedded system, you can add those to the `FDCONFIG.SYS` file. For example, you might need to set environment variables using the `SET` action, or tune the FreeDOS kernel with `FILES=` or `BUFFERS=` statements.

Run the embedded system

With the embedded system fully defined, we can now reboot the machine to run the embedded application. Running an embedded system usually requires only limited resources, so for this demonstration, we'll tweak the QEMU command line to only boot from the hard drive (`-boot order=c`) and not define a CD-ROM drive:

```
qemu-system-i386 -m 8 -hda tiny.img -boot order=c
```

When the FreeDOS kernel starts up, it reads the `FDCONFIG.SYS` file for its startup parameters. Then it runs the shell using the `SHELL=` line. That runs the Simple Senet game automatically.



Running Simple Senet as an embedded system
(Jim Hall, [CC-BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

We've used Simple Senet to demonstrate how to set up an embedded system on FreeDOS. Depending on your needs, you can use whatever standalone application you like. Define it as the DOS shell using the SHELL= line in FDCONFIG.SYS and FreeDOS will automatically launch the application at boot-time.

However, there's one limitation here. Embedded systems do not usually need to exit back to a command prompt, so these dedicated applications don't usually allow the user to quit to DOS. If you manage to exit the embedded application, you'll likely see a "Bad or missing Command Interpreter" prompt, where you'll need to enter the full path to a new shell. For a user-focused desktop system, this would be a problem. But on an embedded system that's dedicated to doing only one job, you should never need to exit anyway.

FreeDOS commands you need to know

By Kevin O'Brien

[FreeDOS](#), the open source implementation of DOS, provides a lightweight operating system for running legacy applications on modern hardware (or in an emulator) and for updating hardware vendor fails with a Linux-compatible firmware flasher. Getting familiar with FreeDOS is not only a fun throwback to the computing days of the past, it's an investment into gaining useful computing skills. In this article, I'll look at some of the essential commands you need to know to work on a FreeDOS system.

Essential directory and file commands

FreeDOS uses directories to organize files on a hard drive. That means you need to use directory commands to create a structure to store your files and find the files you've stored there. The commands you need to manage your directory structure are relatively few:

- **MD** (or **MKDIR**) creates a new directory or subdirectory.
- **RD** (or **RMDIR**) removes (or deletes) a directory or subdirectory.
- **CD** (or **CHDIR**) changes from the current working directory to another directory.
- **DELTREE** erases a directory, including any files or subdirectories it contains.
- **DIR** lists the contents of the current working directory.

Because working with directories is central to what FreeDOS does, all of these (except **DELTREE**) are internal commands contained within **COMMAND.COM**. Therefore, they are loaded into RAM and ready for use whenever you boot (even from a boot disk). The first three commands have two versions: a two-letter short name and a long name. There is no difference in practice, so I'll use the short form in this article.

Make a directory with MD

FreeDOS's `MD` command creates a new directory or subdirectory. (Actually, since the *root* is the main directory, all directories are technically subdirectories, so I'll refer to *subdirectories* in all examples.) An optional argument is the path to the directory you want to create, but if no path is included, the subdirectory is created in the current working subdirectory.

For example, to create a subdirectory called `letters` in your current location:

```
C:\HOME\>MD LETTERS
```

This creates the subdirectory `C:\letters`.

By including a path, you can create a subdirectory anywhere:

```
C:\>MD C:\HOME\LETTERS\LOVE
```

This has the same result as moving into `C:\HOME\LETTERS` first and then creating a subdirectory there:

```
C:\CD HOME\LETTERS
C:\HOME\LETTERS\>MD LOVE
C:\HOME\LETTERS\>DIR
LOVE
```

A path specification cannot exceed 63 characters, including backslashes.

Remove a directory with RD

FreeDOS's `RD` command removes a subdirectory. The subdirectory must be empty. If it contains files or other subdirectories, you get an error message. This has an optional path argument with the same syntax as `MD`.

You cannot remove your current working subdirectory. To do that, you must `CD` to the parent subdirectory and then remove the undesired subdirectory.

Delete files and directories with DELTREE

The `RD` command can be a little confusing because of safeguards FreeDOS builds into the command. That you cannot delete a subdirectory that has contents, for instance, is a safety measure. `DELTREE` is the solution.

DELTREE deletes an entire subdirectory "tree" (a subdirectory), plus all of the files it contains, plus all of the subdirectories those contain, and all of the files *they* contain, and so on, all in one easy command. Sometimes it can be a little *too* easy because it can wipe out so much data so quickly. It ignores file attributes, so you can wipe out hidden, read-only, and system files without knowing it.

You can even wipe out multiple trees by specifying them in the command. This would wipe out both of these subdirectories in one command:

```
C:\>DELTREE C:\F00 C:\BAR
```

This is one of those commands where you really ought to think twice before you use it. It has its place, definitely. I can still remember how tedious it was to go into each subdirectory, delete the individual files, check each subdirectory for contents, delete each subdirectory one at a time, then jump up one level and repeat the process. **DELTREE** is a great timesaver when you need it. But I would never use it for ordinary maintenance because one false move can do so much damage.

Format a hard drive

The **FORMAT** command can also be used to prepare a blank hard drive to have files written to it. This formats the D: drive:

```
C:\>FORMAT D:
```

Copy files

The **COPY** command, as the name implies, copies files from one place to another. The required arguments are the file to be copied and the path and file to copy it to. Switches include:

- /Y prevents a prompt when a file is being overwritten.
- /-Y requires a prompt when a file is being overwritten.
- /V verifies the contents of the copy.

This copies the file **MYFILE.TXT** from the working directory on C: to the root directory of the D: drive and renames it **EXAMPLE.TXT**:

```
C:\>COPY MYFILE.TXT D:\EXAMPLE.TXT
```


This copies the file `EXAMPLE.TXT` from the working directory on `C:` to the `C:\DOCS\` directory and then verifies the contents of the file to ensure that the copy is complete:

```
C:\>COPY EXAMPLE.TXT C:\DOCS\EXAMPLE.TXT /V
```

You can also use the `COPY` command to combine and append files. This combines the two files `MYFILE1.TXT` and `MYFILE2.TXT` and places them in a new file called `MYFILE3.TXT`:

```
C:\>COPY MYFILE1.TXT+MYFILE2.TXT MYFILE3.TXT
```

Copy directories with XCOPY

The `XCOPY` command copies entire directories, along with all of their subdirectories and all of the files contained in those subdirectories. Arguments are the files and path to be copied and the destination to copy them to. Important switches are:

- `/S` copies all files in the current directory and any subdirectory within it.
- `/E` copies subdirectories, even if they are empty. This option must be used with the `/S` option.
- `/V` verifies the copies that were made.

This is a very powerful and useful command, particularly for backing up directories or an entire hard drive.

This command copies the entire contents of the directory `C:\DOCS`, including all subdirectories and their contents (except empty subdirectories) and places them on drive `D:` in the directory `D:\BACKUP\DOCS\`:

```
C:\>XCOPY C:\DOCS D:\BACKUP\DOCS\ /S
```

Using FreeDOS

FreeDOS is a fun, lightweight, open source operating system. It provides lots of great utilities to enable you to get work done on it, whether you're using it to update the firmware of your motherboard or to give new life to an old computer. Learn the basics of FreeDOS. You might be surprised at how versatile it is.

Understanding file names and directories in FreeDOS

By Kevin O'Brien

The open source operating system [FreeDOS](#) is a tried-and-true project that helps users play retro games, update firmware, run outdated but beloved applications, and study operating system design. FreeDOS offers insights into the history of personal computing (because it implements the de facto operating system of the early '80s) but in a modern context. In this article, I'll use FreeDOS to explain how file names and extensions developed.

Understanding file names and ASCII text

FreeDOS file names follow what is called the *8.3 convention*. This means that all FreeDOS file names have two parts that contain up to eight and three characters, respectively. The first part is often referred to as the *file name* (which can be a little confusing because the combination of the file name and the file extension is also called a file name). This part can have anywhere from one to eight characters in it. This is followed by the *extension*, which can have from zero to three characters. These two parts are separated by a dot.

File names can use any letter of the alphabet or any numeral. Many of the other characters found on a keyboard are also allowed, but not all of them. That's because many of these other characters have been assigned a special use in FreeDOS. Some of the characters that can appear in a FreeDOS file name are:

```
~ ! @ # $ % ^ & ( ) _ - { } `
```

There are also characters in the extended [ASCII](#) set that can be used, such as ■.

Characters with a special meaning in FreeDOS that, therefore, cannot be used in file names include:

```
* / + | \ = ? [ ] ; : " . < > ,
```

Also, you cannot use a space in a FreeDOS file name. The FreeDOS console [uses spaces to separate commands](#) from options and parameters.

FreeDOS is case *insensitive*, so it doesn't matter whether you use uppercase or lowercase letters. All letters are converted to uppercase, so your files end up with uppercase letters in the name, no matter what you do.

File extensions

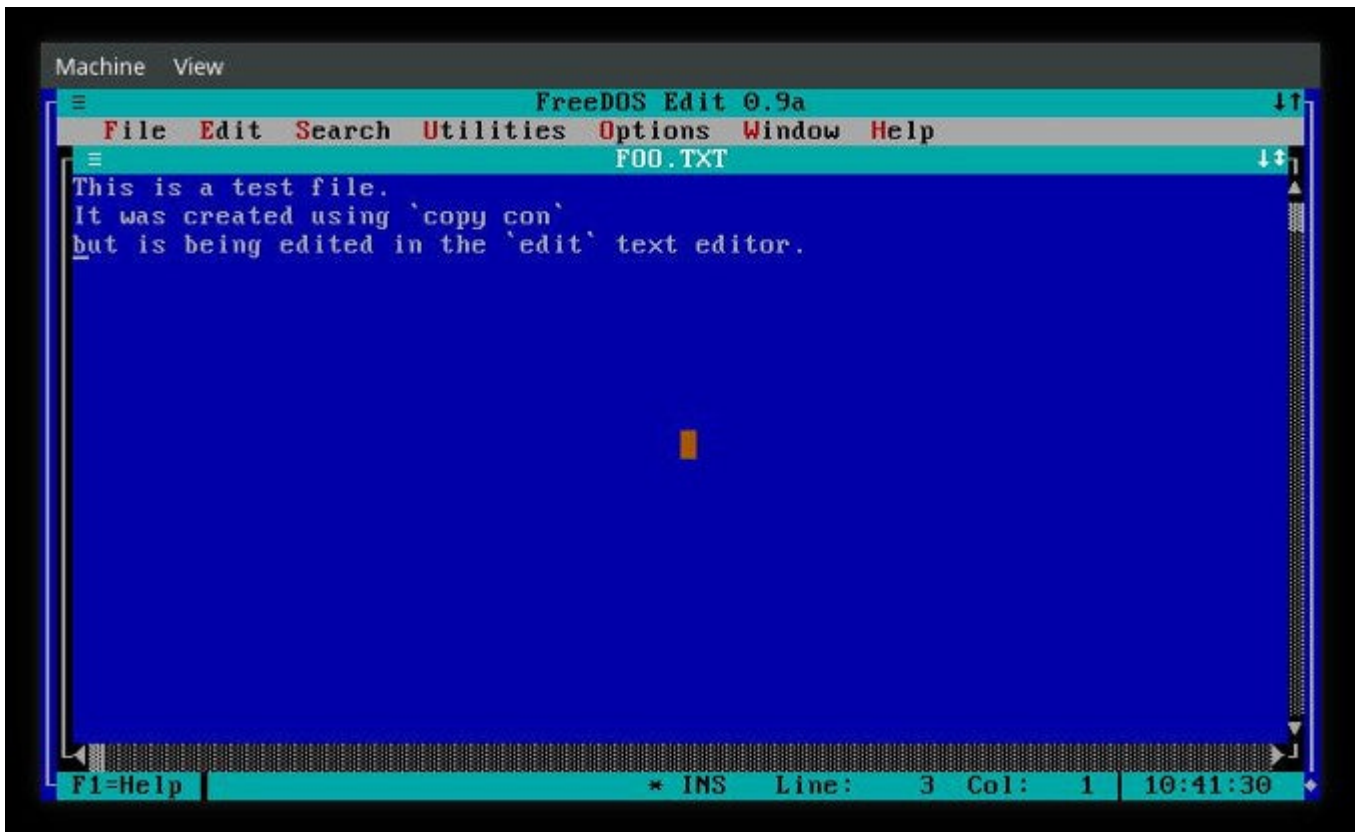
A file in FreeDOS isn't required to have an extension, but file extensions do have some uses. Certain file extensions have built-in meanings in FreeDOS, such as:

- **EXE**: executable file
- **COM**: command file
- **SYS**: system file
- **BAT**: batch file

Specific software programs use other extensions, or you can use them when you create a file. These extensions have no absolute file associations, so if you use a FreeDOS word processor, it doesn't matter what extension you use for your files. You could get creative and use extensions as part of your filing system if you want. For instance, you could name your memos using *.JAN, *.FEB, *.MAR, *.APR, and so on.

Editing files

FreeDOS comes with the Edit application for quick and easy text editing. It's a simple editor with a menu bar along the top of the screen for easy access to all the usual functions (such as copy, paste, save, and so on.)



As you might expect, many other text editors are available, including the tiny but versatile [e3 editor](#). You can find a good variety of [FreeDOS applications](#) on GitLab.

Creating files

You can create empty files in FreeDOS using the `touch` command. This simple utility updates a file's modification time or creates a new file:

```
C:\>touch foo.txt
C:\>dir
FOO      TXT      0  01-12-2021 10:00a
```

You can also create a file directly from the FreeDOS console without using the Edit text editor. First, use the `copy` command to copy input in the console (`con` for short) into a new file object. Terminate input with **Ctrl+Z** followed by the **Return** or **Enter** key:

```
C:\>copy con test.txt
con => test.txt
```

```
This is a test file.  
^Z
```

The **Ctrl+Z** character shows up in the console as ^Z. It isn't copied to the file but serves as an End of File (EOF) delimiter. In other words, it tells FreeDOS when to stop copying. This is a neat trick for making quick notes or starting a simple document to work on later.

Files and FreeDOS

FreeDOS is open source, free, and [easy to install](#). Exploring how FreeDOS treats files can help you understand how computing has developed over the years, regardless of your usual operating system. Boot up FreeDOS and start exploring modern retro computing!

How to navigate FreeDOS with CD and DIR

By Jim Hall

FreeDOS is an open source DOS-compatible operating system that you can use to play classic DOS games, run legacy business software, or develop embedded systems. Any program that works on MS-DOS should also run on FreeDOS.

But if you've never used DOS, you might be confused about how to navigate the system. FreeDOS is primarily a command-line interface; there is no default graphical user interface (GUI) in FreeDOS. You need to type every command at the command line.

Two commands that help you find your way around FreeDOS: **CD** and **DIR**. I've written those commands in all uppercase, but DOS is actually *case insensitive*, so you can type your commands using either uppercase or lowercase letters. DOS doesn't care.

Let's start with the **DIR** command. This command name is short for *directory* and is similar to the `ls` command on Linux systems. You can run **DIR** anywhere on your system to see what files you have. Just type the command **DIR** to get a list of files and directories:

```

D:\>dir
Volume in drive D is FD13-LiveCD

Directory of D:\

DEVEL                <DIR>    04/30/2021  9:43a
FDOS_X86             <DIR>    04/30/2021  9:43a
FREEDOS              <DIR>    04/30/2021  9:37a
GAMES                <DIR>    04/30/2021  9:43a
ISOLINUX             <DIR>    04/30/2021  9:43a
PACKAGES             <DIR>    04/30/2021  9:43a
UTIL                 <DIR>    04/30/2021  9:43a
COMMAND.COM          85,048  04/30/2021  8:54a
FDAUTO.BAT           3,700   04/30/2021  9:42a
FDCONFIG.SYS         182     04/30/2021  9:42a
KERNEL.SYS          46,685  04/30/2021  8:54a
SETUP.BAT            6,744   04/30/2021  8:54a
                    5 file(s)    142,359 bytes
                    7 dir(s)      0 bytes free
D:\>

```

(Jim Hall, CC-BY SA 4.0)

The output from `DIR` is very utilitarian. At the top, `DIR` prints the "volume name" of the current drive. Then `DIR` shows all the files and directories. In the screenshot, you can see the directory listing of the FreeDOS LiveCD. It contains several directories, including the `FREEDOS` directory which contains all of the core FreeDOS programs and utilities. You can also see several files, starting with the `COMMAND.COM` shell, which is similar to Bash on Linux—except much simpler. The FreeDOS kernel itself is the `KERNEL.SYS` file further down the list.

At the top level of any drive, before you go into a directory, you are at the *root directory*. DOS uses the `\` ("back slash") character to separate directories in a path, which is slightly different from the `/` ("slash") character in Linux systems.

To navigate into a directory, you can use the `CD` command. Like `cd` on Linux, this stands for *change directory*. The `CD` command sets the new *working directory* to wherever you want to go. For example, you might go into the `GAMES` directory and use `DIR` to list its contents:

```
FDOS_X86      <DIR> 04/30/2021  9:43a
FREEDOS      <DIR> 04/30/2021  9:37a
GAMES        <DIR> 04/30/2021  9:43a
ISOLINUX     <DIR> 04/30/2021  9:43a
PACKAGES     <DIR> 04/30/2021  9:43a
UTIL         <DIR> 04/30/2021  9:43a
COMMAND  COM  85,048 04/30/2021  8:54a
FDAUTO  BAT   3,700 04/30/2021  9:42a
FDCONFIG SYS  182  04/30/2021  9:42a
KERNEL  SYS  46,685 04/30/2021  8:54a
SETUP   BAT   6,744 04/30/2021  8:54a
      5 file(s)      142,359 bytes
      7 dir(s)       0 bytes free
D:\>cd games
D:\GAMES>dir
Volume in drive D is FD13-LiveCD

Directory of D:\GAMES

.                <DIR> 04/30/2021  9:43a
..               <DIR> 04/30/2021  9:43a
FLPYBIRD        <DIR> 04/30/2021  9:43a
      0 file(s)      0 bytes
      3 dir(s)       0 bytes free
D:\GAMES>
```

(Jim Hall, CC-BY SA 4.0)

You can also specify a path to CD, to jump to a specific directory elsewhere on your system. If I wanted to change to the FREEDOS directory, I could simply specify the full path relative to the root directory. In this case, that's the \FREEDOS directory. From there, I can run another DIR command to see the files and directories stored there:


```
.                <DIR> 04/30/2021  9:43a
..               <DIR> 04/30/2021  9:43a
FLPYBIRD        <DIR> 04/30/2021  9:43a
      0 file(s)          0 bytes
      3 dir(s)           0 bytes free
D:\GAMES>cd \freedos
D:\FREEDOS>dir
Volume in drive D is FD13-LiveCD

Directory of D:\FREEDOS

.                <DIR> 04/30/2021  9:37a
..               <DIR> 04/30/2021  9:43a
APPINFO         <DIR> 04/30/2021  9:43a
BIN             <DIR> 04/30/2021  9:43a
CPI             <DIR> 04/30/2021  9:42a
DOC             <DIR> 04/30/2021  9:43a
HELP           <DIR> 04/30/2021  9:43a
LINKS          <DIR> 04/30/2021  9:43a
NLS            <DIR> 04/30/2021  9:37a
PACKAGES       <DIR> 04/30/2021  9:43a
SETUP          <DIR> 04/30/2021  9:38a
      0 file(s)          0 bytes
      11 dir(s)         0 bytes free
D:\FREEDOS>
```

(Jim Hall, CC-BY SA 4.0)

Like Linux, DOS also uses `.` and `..` to represent a *relative path*. The `.` directory is the current directory, and `..` is the directory that's one level before it, or the *parent* directory. Using `..` allows you to "back up" one directory with the `CD` command, so you don't need to specify a full path.

From the first `DIR` screenshot, we can see the root directory also contains a `DEVEL` directory. If we're already in the `\FREEDOS` directory, we can navigate to `DEVEL` by "backing up" one directory level, and "going into" the `..\DEVEL` directory via a relative path:

```
.          <DIR> 04/30/2021  9:37a
..         <DIR> 04/30/2021  9:43a
APPINFO   <DIR> 04/30/2021  9:43a
BIN       <DIR> 04/30/2021  9:43a
CPI       <DIR> 04/30/2021  9:42a
DOC       <DIR> 04/30/2021  9:43a
HELP      <DIR> 04/30/2021  9:43a
LINKS     <DIR> 04/30/2021  9:43a
NLS       <DIR> 04/30/2021  9:37a
PACKAGES  <DIR> 04/30/2021  9:43a
SETUP     <DIR> 04/30/2021  9:38a
          0 file(s)          0 bytes
          11 dir(s)         0 bytes free
D:\FREEDOS>cd ..\devel
D:\DEVEL>dir
Volume in drive D is FD13-LiveCD

Directory of D:\DEVEL

.          <DIR> 04/30/2021  9:43a
..         <DIR> 04/30/2021  9:43a
MPX       <DIR> 04/30/2021  9:43a
          0 file(s)          0 bytes
          3 dir(s)         0 bytes free
D:\DEVEL>
```

(Jim Hall, CC-BY SA 4.0)

Armed with just two commands DIR and CD, you can navigate your FreeDOS system from the command line. Try it on your FreeDOS system to locate files and execute programs.

FreeDOS commands for Linux fans

By Jim Hall

If you've tried FreeDOS, you might have been stymied by the command line. The DOS commands are slightly different from how you might use the Linux command line, so getting around on the command line requires learning a few new commands.

But it doesn't have to be an "all new" experience for Linux users. We've always included some standard Unix commands in FreeDOS, in addition to the DOS commands that are already similar to Linux. So if you're already familiar with the Linux command line, try these commands to help ease into FreeDOS:

Getting Around

Use the `cd` command to *change directory* in the FreeDOS filesystem. The usage is basically the same on FreeDOS as it is on Linux. To change into a subdirectory called `apps`, type `cd apps`. To go back to the previous directory, type `cd . . .`

The only difference when navigating through directories and paths is that on FreeDOS, the directory separator is `\` ("backslash") instead of `/` ("forward slash") that you use on Linux. For example, let's say you were in the `\devel` directory and you wanted to move to the `\fdos` directory. Both of those are at the same "level" relative to the *root* directory. So you could type `cd . . \fdos` to "back up" one directory level (with `. .`) and then "go into" the `fdos` directory.

To change to a new directory, you could instead give the full path with the leading backslash. This is handy if you are already deep into another path, and just want to switch immediately to the new location. For example, to change to the `\temp` directory, you can type `cd \temp`.

```
C:\>cd apps
C:\APPS>cd ..
C:\>cd devel
C:\DEVEL>cd ..\fdos
C:\FDOS>cd \temp
C:\TEMP>_
```

In FreeDOS, like most DOS systems, you can see your current path as part of the DOS prompt. On Linux, your prompt is probably something like `$`. On FreeDOS, the prompt lists the current drive, the current path within that drive, then `>` as the prompt (taking the place of `$` on Linux).

Listing and Displaying Files

On Linux, the standard command to list files in the current directory is the `ls` command. On FreeDOS, it's a different command: `dir`. But you can get a similar behavior as `ls` by creating an *alias*.

To create an alias to another command, use the built-in `alias` command. For example, use this command to define an alias for `ls` that will display a directory listing in a similar way to using `ls` on Linux:

```
C:\>alias ls=dir /one /w /b /l
C:\>ls
[apps]    command.com    [devel]  fdauto.bat    fdconfig.sys
[fdos]    kernel.sys    [src]    [temp]
```

The command option format is slightly different on FreeDOS than on Linux. On Linux, you start options with a hyphen character (`-`). But on FreeDOS, options start with a forward slash. The `alias` command above uses the slash character—those are options to `dir`. The `/one` option tells `dir` to order (o) in a certain way: sort any files and directories by name (n) and then by extension (e). Using `/w` says to use a "wide" directory listing, `/b` uses a "bare" display without the other information `dir` usually provides, and `/l` instructs `dir` to display files and directories in lowercase.

Note that the command-line options for the FreeDOS `dir` command are quite different from the options to Linux `ls`, so you can't use this `ls` alias exactly like you would on Linux. For

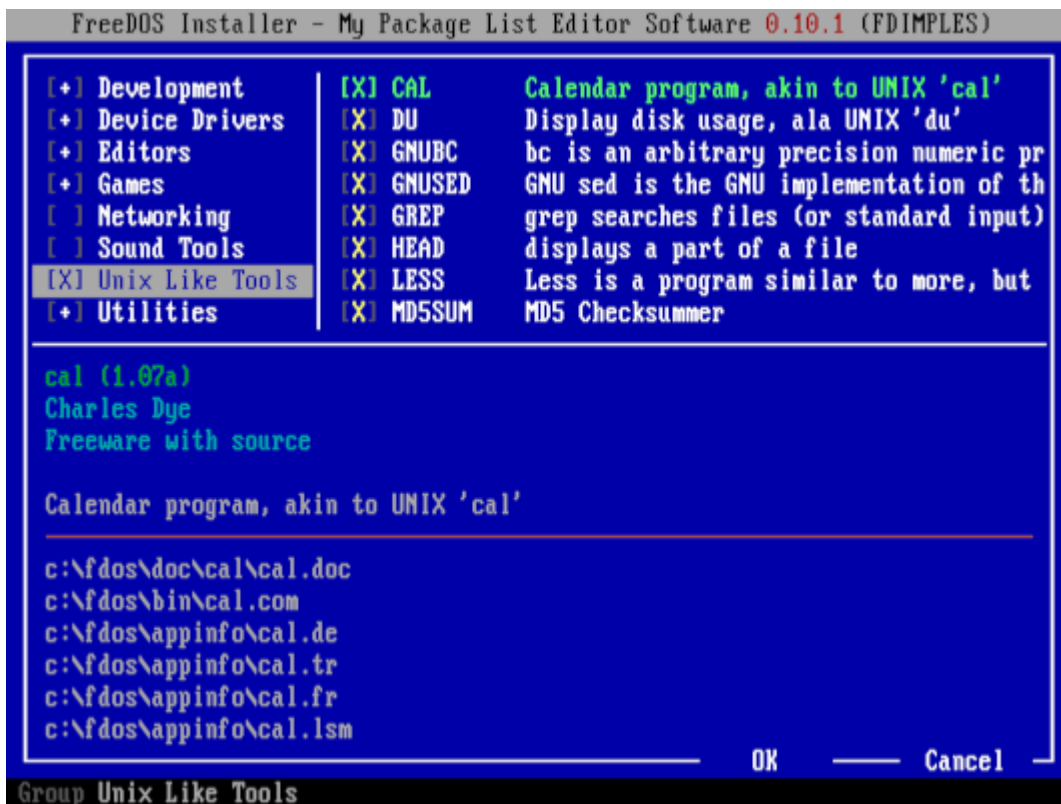
example, typing `ls -l` with this alias on FreeDOS will result in a "File not found" error, because the underlying FreeDOS `dir` command will be unable to find a file called `-l`. But for basic "see what files I have on my system," this `ls` alias is good enough to help Linux users get started with FreeDOS.

Similarly, you can create an alias for the FreeDOS `type` command, to act like the Linux `cat` command. Both programs display the contents of a text file. While `type` doesn't support the command-line options you might use under Linux, the basic usage to display a single file will be the same.

```
C:\FDOS>alias cat=type
C:\FDOS>cat version.fdi
PLATFORM=FreeDOS
VERSION=1.3-RC4
RELEASE=2021-04-30
C:\FDOS>
```

Other Unix-like Commands

FreeDOS includes a selection of other common Unix-like commands, so Linux users will feel more at home. To use these Linux commands on FreeDOS, you may need to install the **Unix Like Tools** package from the **FreeDOS Installer - My Package List Editor Software** (FDIMPLES) package manager.



(Jim Hall, CC-BY SA 4.0)

Not all of the Unix-like utilities work *exactly* like their Linux counterparts. That's why we call them *Unix-like*. You might want to check the compatibility if you're using some esoteric command-line options, but typical usage should be fine. Start with these common Unix-like commands on FreeDOS:

The `cal` command is the standard Unix calendar program. For example, to display the calendar for the current month, just type `cal`. To view a specific month, give the month and year as arguments:

```
C:\>cal 6 1994
    June 1994
Su Mo Tu We Th Fr Sa
           1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
```

View your disk usage with the `du` command. This is a simple version of the Linux *disk usage* command and doesn't support any command-line options other than a path.

```
C:\>du -s apps
usage: du (start path)
C:\>du apps
  158784 C:\APPS\FED
      0 C:\APPS
Total from C:\APPS is 158784
C:\>
```

The `head` command displays the first few lines of a file. For example, this is a handy way to determine if a file contains the correct data.

```
C:\>head fdauto.bat
@ECHO OFF
set DOSDIR=C"\FDOS
set LANG=EN
set TZ=UTC
set PATH=%dosdir%\BIN
if exist %dosdir%\LINKS\NUL set PATH=%path%;%dosdir%\LINKS
set NLSPATH=%dosdir%\NLS
set HELPPATH=%dosdir%\HELP
set TEMP=%dosdir%\TEMP
set TMP=%TEMP%
C:\>
```

To view an entire file, use the `more` command, the default file viewer on FreeDOS. This displays a file one screenful at a time, then prints a prompt to press a key before displaying the next screenful of information. The `more` command is a very simple file viewer; for a more full-featured viewer like you might use on Linux, try the `less` command. The `less` command provides the ability to scroll "backwards" through a file, in case you missed something. You can also search for specific text.

```
C:\>less fdauto.bat
@ECHO OFF
set DOSDIR=C"\FDOS
set LANG=EN
set TZ=UTC
set PATH=%dosdir%\BIN
if exist %dosdir%\LINKS\NUL set PATH=%path%;%dosdir%\LINKS
set NLSPATH=%dosdir%\NLS
set HELPPATH=%dosdir%\HELP
set TEMP=%dosdir%\TEMP
set TMP=%TEMP%
[...]
```

If you have a lot of directories in your program path variable (**PATH**) and aren't sure where a certain program is running from, you can use the **which** command. This scans the program path variable, and prints the full location of the program you are looking for.

```
C:\>which less
less    C:\>FDOS\BIN\LESS.EXE
C:\>_
```

FreeDOS includes other Unix-like commands that you might use in other, more specific situations. These include:

- **bc**: Arbitrary precision numeric processing language
- **sed**: Stream editor
- **grep** and **xgrep**: Search a text file using regular expression
- **md5sum**: Generate an MD5 signature of a file
- **nro**: Simple typesetting using nroff macros
- **sleep**: Pause the system for a few seconds
- **tee**: Save a copy of a command-line stream
- **touch**: Modify a file's timestamp
- **trch**: Translate single characters (like Linux tr)
- **uptime**: Report how long your FreeDOS system has been running

FreeDOS at your command

FreeDOS, like Linux and BSD, is open source. Whether you want to challenge yourself by learning a new style of command-line interaction, or you want to fall back on the comfort of familiar Unix-like tools, FreeDOS is a fun and fresh operating system to explore. Give it a try!

Set your path in FreeDOS

By Kevin O'Brien

Everything you do in the open source [FreeDOS](#) operating system is done from the command line. The command line begins with a *prompt*, which is the computer's way of saying, "I'm ready. Give me something to do." You can configure your prompt's appearance, but by default, it's:

```
C:\>
```

From the command line, you can do two things: Run an internal command or run a program. External commands are programs found in separate files in your `FDOS` directory, so running programs includes running external commands. It also means running the application software you use to do things with your computer. You can also run a batch file, but in that case, all you're doing is running a series of commands or programs that are listed in the batch file.

Executable application files

FreeDOS can run three types of application files:

1. **COM** is a file in machine language less than 64KB in size.
2. **EXE** is a file in machine language that can be larger than 64KB. EXE files also have information at the beginning of the file telling DOS what type of file it is and how to load and run it.
3. **BAT** is a *batch file* written with a text editor in ASCII text format containing FreeDOS commands that are executed in batch mode. This means each command is executed in sequence until the file ends.

If you enter an application name that FreeDOS does not recognize as either an internal command or a program, you get the error message *Bad command or filename*. If you see this error, it means one of three things:

1. The name you gave is incorrect for some reason. Possibly you misspelled the file name, or maybe you're using the wrong command name. Check the name and the spelling and try again.
2. Maybe the program you are trying to run is not installed on the computer. Verify that it is installed.
3. The file does exist, but FreeDOS doesn't know where to find it.

The final item on this list is the subject of this article, and it's referred to as the **PATH**. If you're used to Linux or Unix already, you may already understand the concept of [the PATH variable](#). If you're new to the command line, the path is an important thing to get comfortable with.

The path

When you enter the name of an executable application file, FreeDOS has to find it. FreeDOS looks for the file in a specific hierarchy of locations:

1. First, it looks in the active directory of the current drive (called the *working directory*). If you're in the directory `C:\FDOS`, and you type in the name `FOOBAR.EXE`, FreeDOS looks in `C:\FDOS` for a file with that name. You don't even need to type in the entire name. If you type in `FOOBAR`, FreeDOS looks for any executable file with that name, whether it's `FOOBAR.EXE`, `FOOBAR.COM`, or `FOOBAR.BAT`. Should FreeDOS find a file matching that name, it runs it.
2. If FreeDOS does not find a file with the name you've entered, it consults something called the **PATH**. This is a list of directories that DOS has been instructed to check whenever it cannot find a file in the current active directory.

You can see your computer's path at any time by using the **PATH** command. Just type `path` at the FreeDOS prompt, and FreeDOS returns your path setting:

```
C:\>path
PATH=C:\FDOS\BIN
```

The first line is the prompt and the command, and the second line is what the computer returned. You can see that the first place DOS looks is `FDOS\BIN`, which is located on the **C** drive. If you want to change your path, you can enter a `path` command and the new path you want to use:

```
C:\>path=C:\HOME\BIN;C:\FDOS\BIN
```

In this example, I set my path to my personal **BIN** folder, which I keep in a custom directory called **HOME**, and then to **FDOS\BIN**. Now when you check your path:

```
C:\>path  
PATH=C:\HOME\BIN;C:\FDOS\BIN
```

The path setting is processed in the order that directories are listed.

You may notice that some characters are lower case and some upper case. It really doesn't matter which you use. FreeDOS is not case-sensitive and treats everything as an upper-case letter. Internally, FreeDOS uses all upper-case letters, which is why you see the output from your commands in upper case. If you type commands and file names in lower case, a converter automatically converts them to upper case, and they are executed.

Entering a new path replaces whatever the path was set to previously.

The autoexec.bat file

The next question you might have is where that first path, the one FreeDOS uses by default, came from. That, along with several other important settings, is defined in the **AUTOEXEC.BAT** file located at the root of your **C** drive. This is a batch file that automatically executes (hence the name) when you start FreeDOS. You can edit this file with the FreeDOS program **EDIT**. To see or edit the contents of this file, enter the following command:

```
C:\>edit autoexec.bat
```

This line appears near the top:

```
SET PATH=%dosdir%\BIN
```

This line defines the value of the default path.

After you look at **AUTOEXEC.BAT**, you can exit the **EDIT** application by pressing the following keys in order:

1. Alt
2. f
3. x

You can also use the keyboard shortcut **Alt+X**.

Using the full path

If you forget to include `C:\FDOS\BIN` in your path, you won't have immediate access to any of the applications stored there because FreeDOS won't know where to find them. For instance, imagine I set my path to my personal collection of applications:

```
C:\>path=C:\HOME\BIN
```

Applications built into the command line still work:

```
C:\cd HOME
C:\HOME>dir
ARTICLES
BIN
CHEATSHEETS
GAMES
DND
```

However, external commands fail:

```
C:\HOME\ARTICLES>BZIP2 -c example.txt
Bad command or filename - "BZIP2"
```

You can always execute a command that you know is on your system but not in your path by providing the *full path* to the file:

```
C:\HOME\ARTICLES>C:\FDOS\BIN\BZIP2 -c example.txt
C:\HOME\ARTICLES>DIR
example.txb
```

You can execute applications from external media or other directories the same way.

FreeDOS path

Generally, you probably want to keep `C:\PDOS\BIN` in your path because it contains all the default applications distributed with FreeDOS.

Unless you change the path in `AUTOEXEC.BAT`, the default path is restored after a reboot.

Now that you know how to manage your path in FreeDOS, you can execute commands and maintain your working environment in whatever way works best for you.

Appendix: Navigate your FreeDOS system

By Kevin O'Brien

[FreeDOS](#) is an open source implementation of DOS. It's not a remix of Linux, and it is compatible with the operating system that introduced many people to personal computing. This makes it an important resource for running legacy applications, playing retro games, updating firmware on motherboards, and experiencing a little bit of living computer history. In this article, I'll look at some of the essential commands used to navigate a FreeDOS system.

Change your current directory with CD

When you first boot FreeDOS, you're "in" the root directory, which is called `C:\`. This represents the foundation of your filesystem, specifically the system hard drive. It's labeled with a `C` because, back in the old days of MS-DOS and PC-DOS, there were always `A` and `B` floppy drives, making the physical hard drive the third drive by default. The convention has been retained to this day in FreeDOS and the operating system that grew out of MS-DOS, Windows.

There are many reasons not to work exclusively in your root directory. First of all, there are limitations to the FAT filesystem that would make that impractical at scale. Secondly, it would make for a very poorly organized filesystem. So it's common to make new directories (or "folders," as we often refer to them) to help keep your work tidy. To access these files easily, it's convenient to change your working directory.

The FreeDOS `CD` command changes your current working subdirectory to another subdirectory. Imagine a computer with the following directory structure:

```
C:\
 \LETTERS\
  \LOVE\
  \BUSINESS\
 \DND\
 \MEMOS\
 \SCHOOL\
```

You start in the `C:\` directory, so to navigate to your love letter directory, you can use `CD`:

```
C:\>CD \LETTERS\LOVE\
```

To navigate to your `\LETTERS\BUSINESS` directory, you must specify the path to your business letters from a common fixed point on your filesystem. The most reliable starting location is `C:\`, because it's where *everything* on your computer is stored.

```
C:\LETTERS\LOVE\>CD C:\LETTERS\BUSINESS
```

Navigating with dots

There's a useful shortcut for navigating your FreeDOS system, which takes the form of dots. Two dots (`..`) tell FreeDOS you want to move "back" or "down" in your directory tree. For instance, the `LETTERS` directory in this example system contains one subdirectory called `LOVE` and another called `BUSINESS`. If you're in `LOVE` currently, and you want to step back and change over to `BUSINESS`, you can just use two dots to represent that move:

```
C:\LETTERS\LOVE\>CD ..\BUSINESS  
C:\LETTERS\BUSINESS\>
```

To get all the way back to your root directory, just use the right number of dots:

```
C:\LETTERS\BUSINESS\>CD ..\..\.  
C:\>
```

Navigational shortcuts

There are some shortcuts for navigating directories, too.

To get back to the root directory from wherever you are:

```
C:\LETTERS\BUSINESS\>CD \  
C:\>
```

List directory contents with DIR

The `DIR` command displays the contents of a subdirectory, but it can also function as a search command. This is one of the most used commands in FreeDOS, and learning to use it properly is a great time saver.

DIR displays the contents of the current working subdirectory, and with an optional path argument, it displays the contents of some other subdirectory:

```
C:\LETTERS\BUSINESS\>DIR
MTG_CARD   TXT   1344 12-29-2020  3:06p
NON        TXT    381 12-31-2020  8:12p
SOMUCHFO   TXT    889 12-31-2020  9:36p
TEST       BAT    32  01-03-2021 10:34a
```

Attributes

With a special attribute argument, you can use DIR to find and filter out certain kinds of files. There are 10 attributes you can specify:

H	Hidden
-H	Not hidden
S	System
-S	Not system
A	Archivable files
-A	Already archived files
R	Read-only files
-R	Not read-only (i.e., editable and deletable) files
D	Directories only, no files
-D	Files only, no directories

These special designators are denoted with /A: followed by the attribute letter. You can enter as many attributes as you like, in order, without leaving a space between them. For instance, to view only hidden directories:

```
C:\MEMOS\>DIR /A:HD
.OBSCURE   <DIR> 01-08-2021 10:10p
```

Listing in order

You can also display the results of your **DIR** command in a specific order. The syntax for this is very similar to using attributes. You leave a space after the **DIR** command or after any other switches, and enter **/O:** followed by a selection. There are 12 possible selections:

N	Alphabetical order by file name
-N	Reverse alphabetical order by file name
E	Alphabetical order by file extension
-E	Reverse alphabetical order by file extension
D	Order by date and time, earliest first
-D	Order by date and time, latest first
S	By size, increasing
-S	By size, decreasing
C	By DoubleSpace compression ratio, lowest to highest (version 6.0 only)
-C	By DoubleSpace compression ratio, highest to lowest (version 6.0 only)
G	Group directories before other files
-G	Group directories after other files

To see your directory listing grouped by file extension:

```
C:\>DIR /O:E
TEST      BAT 01-10-2021 7:11a
TIMER     EXE 01-11-2021 6:06a
AAA       TXT 01-09-2021 4:27p
```

This returns a list of files in alphabetical order of file extension.

If you're looking for a file you were working on yesterday, you can order by modification time:

```
C:\>DIR /O:-D
AAA       TXT 01-09-2021 4:27p
TEST     BAT 01-10-2021 7:11a
TIMER    EXE 01-11-2021 6:06a
```

If you need to clean up your hard drive because you're running out of space, you can order your list by file size, and so on.

Multiple arguments

You can use multiple arguments in a **DIR** command to achieve fairly complex results. Remember that each argument has to be separated from its neighbors by a blank space on each side:


```
C:\>DIR /A:A /O:D /P
```

This command selects only those files that have not yet been backed up (`/A:A`), orders them by date, beginning with the oldest (`/O:D`), and displays the results on your monitor one page at a time (`/P`). So you can really do some slick stuff with the `DIR` command once you've mastered these arguments and switches.

Terminology

In case you were wondering, anything that modifies a command is an argument.

If it has a slash in front, it is a switch. So all switches are also arguments, but some arguments (for example, a file path) are not switches.

Better navigation in FreeDOS

FreeDOS can be very different from what you're used to if you're used to Windows or macOS, and it can be just different enough if you're used to Linux. A little practice goes a long way, though, so try some of these on your own. You can always get a help message with the `/?` switch. The best way to get comfortable with these commands is to practice using them.